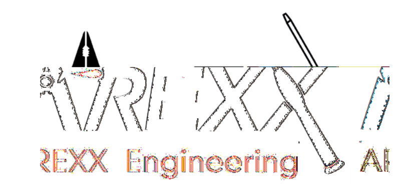
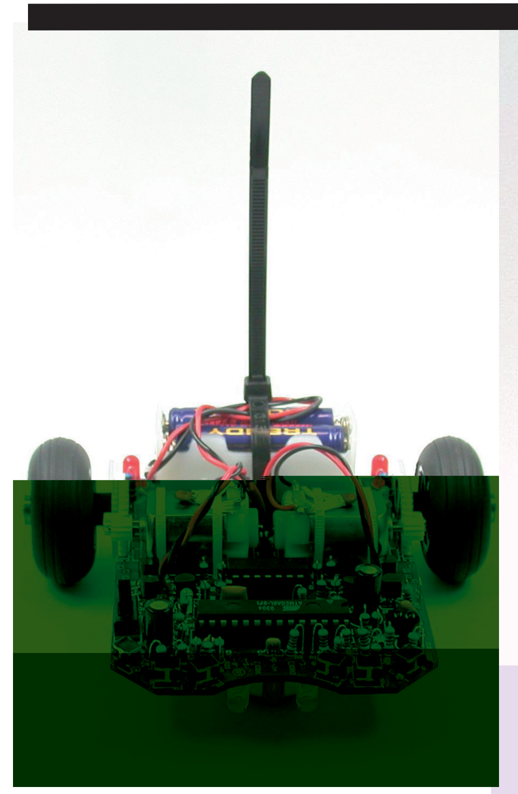
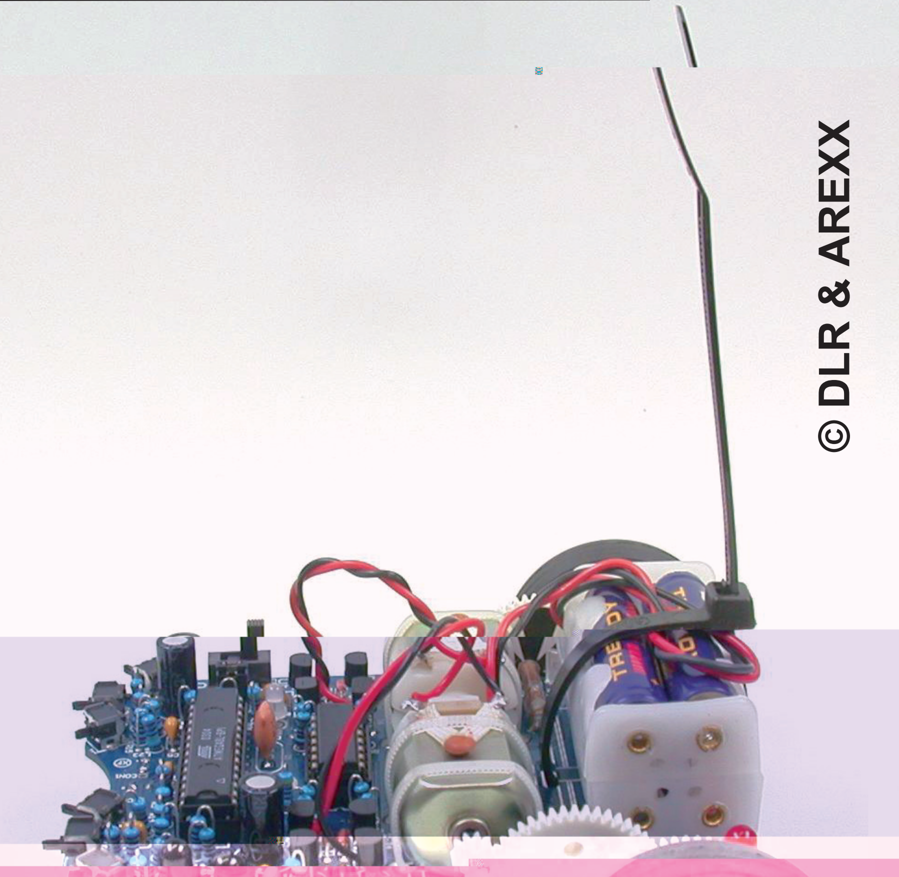
Licence by DLR



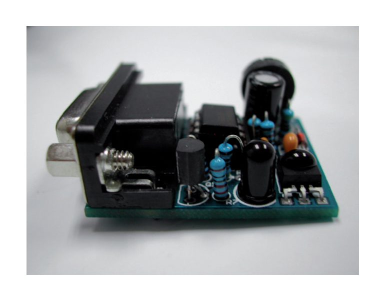


ASURO





© DLR & AREXX





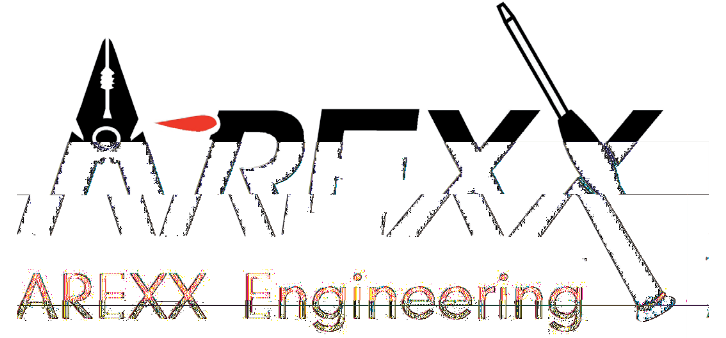
MODE D'EMPLOI

Modèle ARX-03

Fabricant

AREXX, Zwolle - PAYS BAS
JAMA, Taichung - TAIWAN

www.arexx.com



ASURO
Assemblage & Mise en Service

Introduction

ASURO est un petit robot mobile entièrement programmable en C qui a été développé à l'Institut de Robotique et de Mécatronique du Centre allemand d'aéronautique et d'aérospatiale (DLR) à des fins d'enseignement. Pour l'électronicien expérimenté, son assemblage est un jeu d'enfants. Pour le débutant, il est tout à fait à sa portée. A l'exception des circuits imprimés, tous les composants sont des standards disponibles dans le commerce. La programmation se fait uniquement avec des logiciels Freeware. ASURO est donc idéal pour des techniciens amateurs qui souhaitent se lancer dans les circuits commandés par processeur, des projets d'études, des formations continues ou des cours du soir pour adultes. Etant donné que toute l'électronique fait appel à des logiciels Freeware (pour le domaine privé), ASURO apporte la preuve qu'il est tout à fait possible de construire un robot opérationnel sans logiciels, technologie et machines-outils compliqués et coûteux.

ASURO possède en plus de son processeur RISC deux moteurs à commande individuelle, un traceur de ligne optique, six détecteurs de collision, deux odomètres pour les roues, trois afficheurs optiques et une unité de communication à infrarouge qui permet la programmation et la télécommande à partir d'un PC (voir fig. 0.1).

Le point d'exclamation ⚠ dans un triangle attire l'attention sur des consignes qu'il faut respecter scrupuleusement sous peine de destruction du matériel ou de blessures.

Il va de soi que ASURO n'est pas un jouet et est interdit aux enfants de moins de trois ans car il contient des douzaines de petites pièces qu'un enfant peut avaler.
Il ne vous reste plus qu'à préparer une balle de ping-pong et des piles ou des batteries et vous pouvez commencer.

Ah oui, ASURO est le sigle pour « Another Small and Uhh… ? Robot from Oberpfaffenhofen » !

Oberpfaffenhofen 2004

Jan Grewe
Robin Gruber

WWW.DLR.DE

Fig. 0.1 SYNOPTIQUE ASURO (Annexe E)

Transceiver (RS232/USB 2400 Bd)
Serielle IR-Schnittstelle (Programmierung, Kommunikation)
Batterieüberwachung
Status LED (zweifarbig)
Back LEDs
Microcontroller (Atmel ATmega8)
Antriebe
Linienfolge Sensoren (Fototransitor, LED)
Taktile Sensoren (6 unabhängig auswertbare Taster)
Odometrie (Reflexlichtschranke)



Mises en Garde

- ▶ Le droit de retourner le produit s'éteint avec l'ouverture des sachets contenant les pièces et composants.
- ▶ Avant de commencer le montage, lire attentivement le mode d'emploi.
- ▶ Manier les outils avec prudence.
- ▶ Ne pas construire le robot en présence d'enfants en bas âge. Ils peuvent se blesser avec les outils ou avaler de petits composants.
- ▶ Respecter la polarité de la pile.
- ▶ Ne pas mouiller ni la pile, ni le support de pile. Si le ASURO est mouillé, retirer la pile et sécher toutes les pièces au mieux.
- ▶ En cas de non-utilisation pendant plus d'une semaine, il est recommandé de retirer la pile.

Table des matières

Partie I. Mécanique

1. Outils nécessaires

Pour le montage d’ASURO, il vous faut - en plus des composants - les outils et matériaux suivants:

Petit étau ou troisième main car 2 mains ne sont pas toujours suffisantes
Cutter ou scie
Petite pince
Une petite pince coupante de côté pour l’électronique
Eventuellement une pince à dénuder
Fer à souder : Il est recommandé d’utiliser un fer à souder pour l’électronique (env. 20W à 40W) ou une station de soudage (au moins 50W)
Soudure : Brasure pour l’électronique de 1mm d’épaisseur, le cas échéant sans plomb
Fil à dessouder d’env. 2-3mm pour enlever la brasure aux endroits où elle ne doit pas se trouver
Papier de verre à grain fin
Colle rapide, à deux composants ou à chaud
Eventuellement un petit marteau
Eventuellement un multimètre
Ordinateur : Portable ou PC avec Windows ou Linux



Fig. 1.1.: Outillage nécessaire

2. Préparatifs mécaniques

Avant de se lancer dans les travaux, il est impératif de vérifier si toutes les pièces nécessaires sont au complet. Le plus simple est de se servir de la nomenclature des pièces en annexe A. Avant de commencer la partie électronique, il faut effectuer quelques travaux mécaniques.

2.1. Pignons

Pour que les moteurs puissent transmettre leur force à l'engrenage, il faut monter les pignons (ce sont les petites roues dentées présentant une perforation de 1,9mm et 10 dents) sur l'axe moteur. Si les pignons n'ont pas encore été montés sur les moteurs livrés, il faut le faire maintenant. A cet effet, il faut placer un pignon sans forcer sur l'axe de chaque moteur. Il s'agit simplement de le faire tenir sur l'axe. Posez ensuite le moteur avec le pignon vers le bas sur une surface pas trop dure (plastique, carton, etc.) et tapotez doucement avec un petit marteau sur l'axe qui dépasse sur le dessus jusqu'à ce que l'axe est complètement enfoncé dans le pignon (voir fig. 2.1). Il est également possible d'enfoncer le pignon dans l'axe à l'aide d'un étau. Dans ce cas, il faut cependant exercer la pression uniquement sur l'arbre moteur et non pas sur le boîtier ou les roulements.

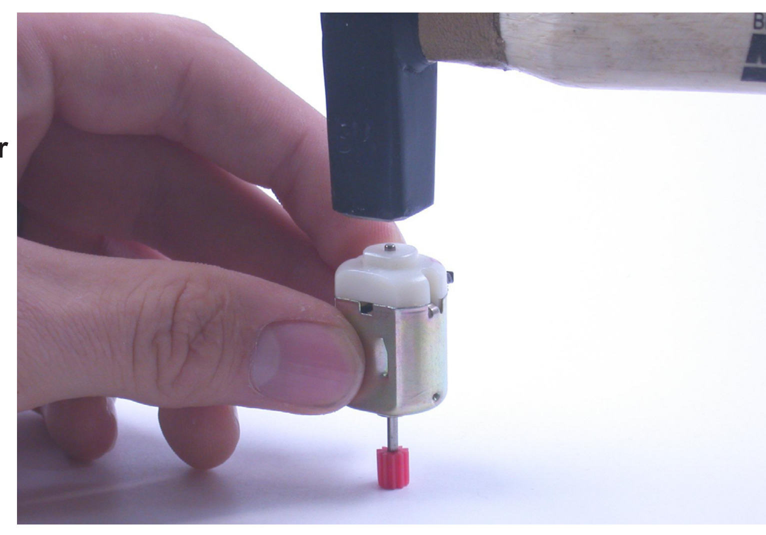


Fig. 2.1.: Montage des pignons sur l'axe moteur

2.2. Balle de ping-pong

ASURO doit glisser plus tard sur une demi-balle de ping-pong qu'il faut fabriquer. Le mieux est de prendre une balle entière et de la scier ou couper en deux à l'aide d'un cutter (attention aux doigts !). Ecrêtez les bords à l'aide d'une lime ou d'un papier de verre.



Fig. 2.2.: Balle de ping-pong sciée en deux

Ne pas utiliser d'outils électriques en raison du risque d'incendie!



2.3. Détecteurs de roue

Pour ne pas décevoir la diode luminescente et le transistor photoélectrique (barrière lumineuse pour l'odométrie) qui se tourneront pas la suite en toute confiance vers le premier engrenage, il faut appliquer sur la face sans pignon des deux premières roues dentées (celles à 50 et à 10 dents) les autocollants à motif noir et blanc (voir fig. 2.3.).

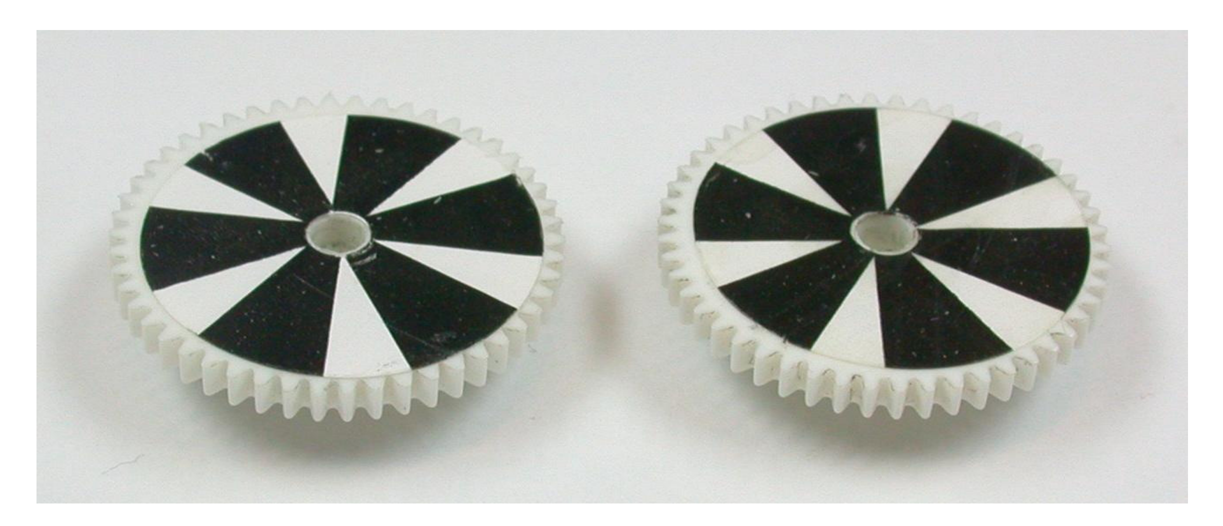


Fig. 2.3.: Montage der Radsensormuster

Plus le motif possède de segments, plus le nombre de tours de la roue dentée et donc la vitesse d'ASURO sont déterminés avec précision. D'un autre côté, la différence mesurée entre clair et sombre diminue.

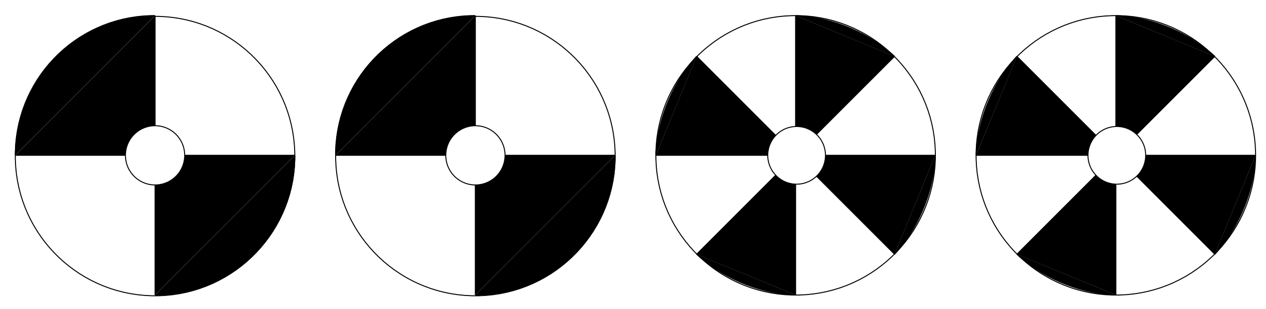


Fig. 2.4.: Exemple de détecteurs

Voilà, la première partie est terminée. Les composants mécaniques sont tous prêts maintenant

Une courte pause...

Et on continue avec l'électronique.

Partie II. Electronique

3. Petit Manuel du Brasage

Bien qu'ASURO ne comporte que des composants câblés et se prête donc à merveille à l'implantation manuelle à la différence des composants SMD implantés en surface (Fig. 3.1 montre la comparaison entre le boîtier le plus petit et le nôtre dans lequel le processeur d'ASURO est disponible. La puce au silicium est la même dans les deux boîtiers !), il faut respecter certaines consignes, surtout pour les soudeurs peu expérimentés.



Il va de soi que le circuit imprimé doit être absolument hors tension. Eteindre ne signifie pas ‚hors tension'! Il faut enlever les piles!

3.1. Panne, Brasage et Température

La Fig. 3.2 montre les bases les plus importantes du brasage!

L'extrémité dangereuse doit avoir une température de brasage d'env. 360°C pour une soudure à teneur en plomb et env. 390°C pour une brasure sans plomb. Pour amorcer le brasage des axes, elle peut être un peu plus élevée (420°C). Il est recommandé d'utiliser une panne pointue comme un crayon de papier pour les implantations sur une platine électronique. Vous pouvez utiliser une panne plus large pour les axes.

N'oubliez pas d'humidifier l'éponge (elle ne doit pas goutter) et d'étamer la panne avec un peu de soudure juste avant d'appliquer le fer à souder sur l'emplacement à braser après une pause ou au début du brasage. Si des résidus de brasure sont restés accrochés à la panne, il faut simplement les essuyer sur l'éponge.

Utilisez de la brasure électronique de 0,8 ou 1mm de diamètre.

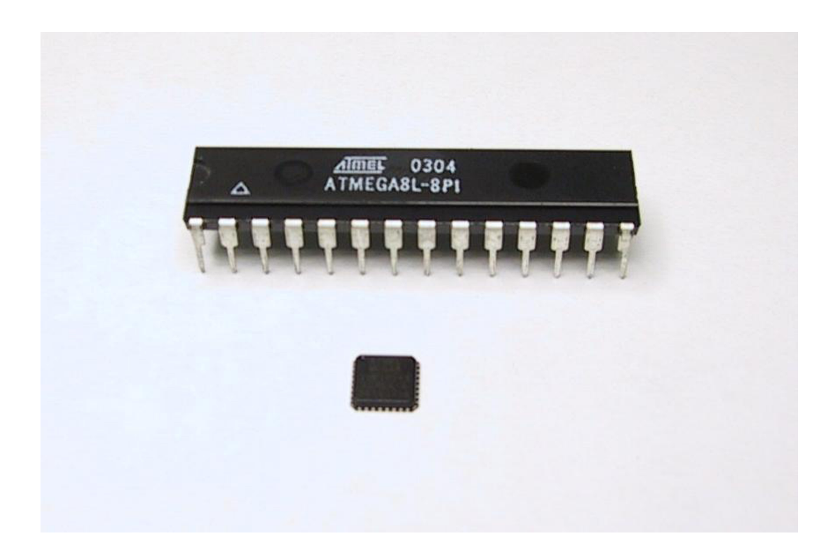


Fig. 3.1.: Comparaison du boîtier le plus grand et le plus petit qui existe pour l'ATmega8L



Fig. 3.2.: Bases du brasage



Des vapeurs qui se forment pendant le brasage sont nocifs à long terme. Evitez de les respirer. Si possible, aspirez-les!
Un autre type de brasure que l'étain de brasage pour l'électronique et de flux à base d'acide risquent de détruire le circuit!

3.2. Préparation des Composants

Toute personne ayant une expérience du brasage électronique, connait le problème : il manque toujours une main. C'est pourquoi ils existent quelques astuces pour maîtriser les différents composants jusqu'à ce que l'on sache se servir correctement d'un fer à souder et de brasure.

Les extrémités de connexion des transistors, diodes luminescentes, circuits imprimés, commutateurs boutons, condensateurs et jumpers d'ASURO sont déjà dirigées dans un sens. Cela reste encore à faire sur les diodes et les résistances.
Pour des raisons de place, toutes les résistances sont implantées à la verticale. Cela signifie que l'un des fils reste tel qu'il est et l'autre doit être courbé à 180°. La courbe devrait avoir un diamètre de 2,5mm et se situer à quelques millimètres de distance de la résistance de façon à ne pas exercer une force mécanique qui risquerait de diminuer son fonctionnement.

Lors du brasage ultérieur, un cercle sur le circuit imprimé indique au-dessus de quel trou la résistance doit se placer et un petit trait dans quel trou il faut introduire le fil courbé.

Les diodes sont implantées à l'horizontale, c'est-à-dire les deux fils doivent être pliés à 90° (si possible au moyen d'une petite pince) à une distance qui permet de les implanter facilement dans les perforations correspondant sur la platine.



Le processeur IC1 ATmega8, le composant IC3 CD4081 et le récepteur IR IC2 SFH5110-36 sont sensibles à l'électricité statique.

Cela signifie qu'ils peuvent être détruits par un simple contact si la personne est chargée électriquement ce qui peut arriver en traversant simplement un tapis. Avant de manipuler ces composants, il est conseillé de se relier à la terre au moyen d'un ruban de masse ou tout du moins toucher le boîtier métallique d'un appareil ou un radiateur.

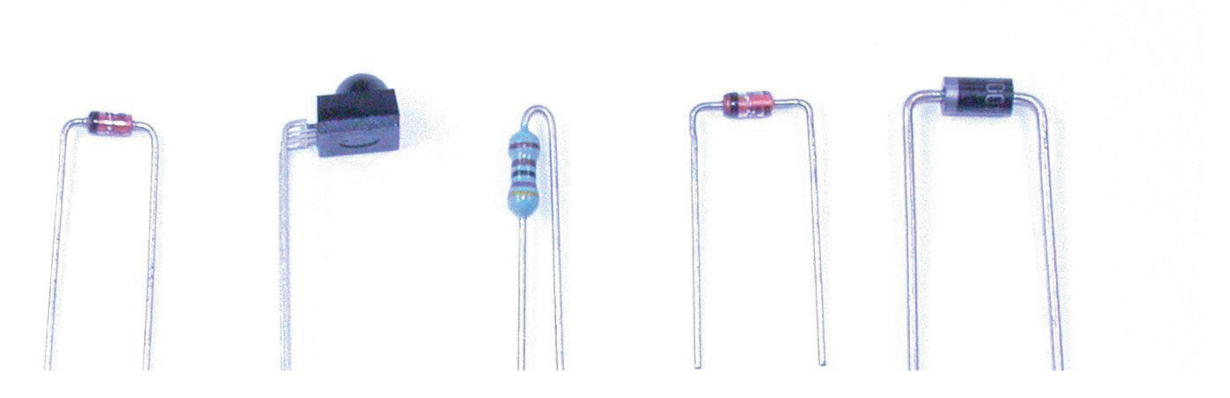


Fig. 3.3.: Composants avec des fils pliés correctement

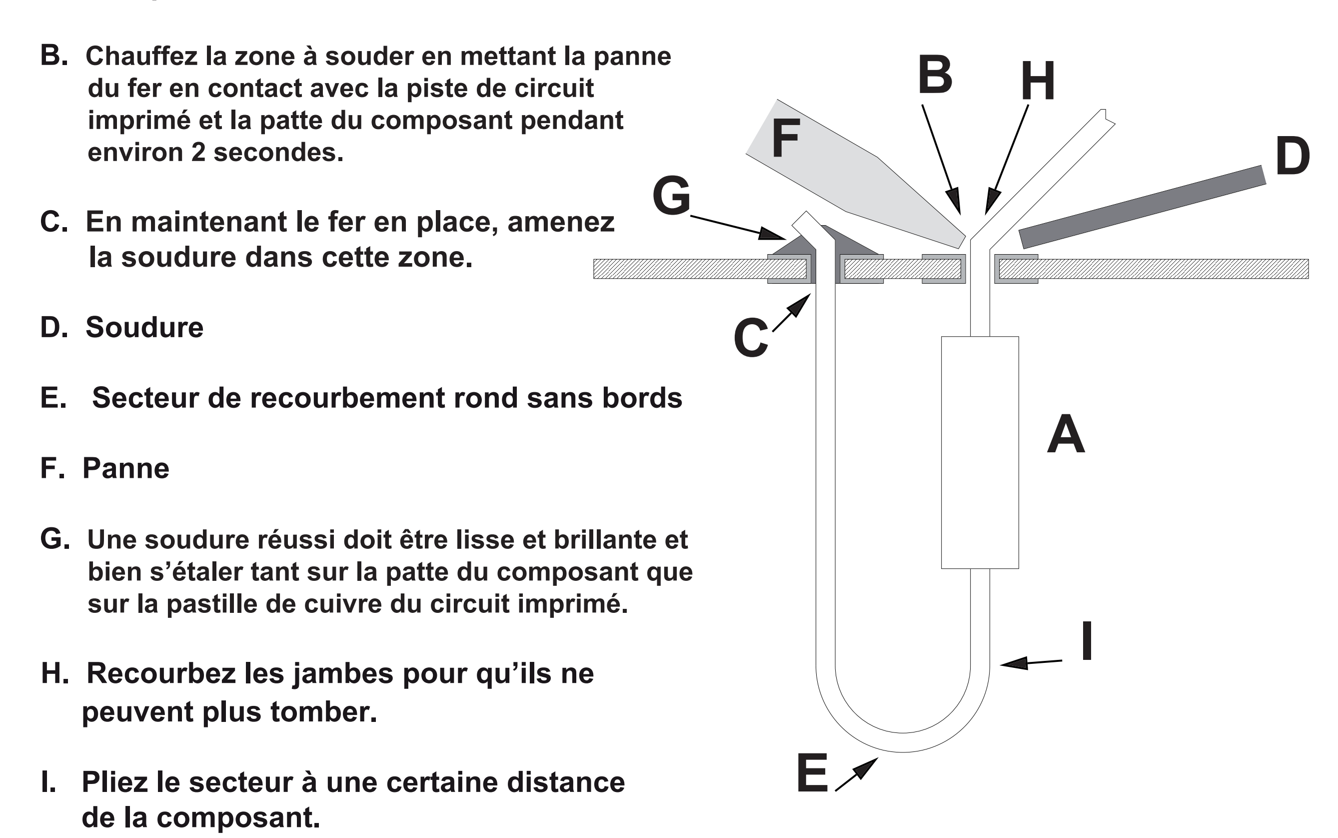


Fig. 3.4.:
Réalisation d'un point de brasure propre

3.3. Brasage des composants

Lorsque les composants ont été préparés, il faut les passer dans les perforations sur la platine. Pour les composants n'ayant que 2 ou 3 fils, pliez les fils en dessous de la platine directement à l'endroit de la perforation (env. 30 à 40° sont suffisants) pour qu'ils ne peuvent plus tomber.

Quant aux composants possédant plus de fils tels que les sockets pour les circuits imprimés, il suffit de plier les deux fils opposés vers l'extérieur. Ne pliez pas les fils à plus de 45° car en cas d'erreur, il devient très difficile de les retirer.

Lorsque le composant a été fixé, il faut chauffer simultanément le fil et la pastille au moyen de la panne du fer à souder, en ajoutant un peu de fil de soudure à cet endroit. Celui-ci fusionne et s'écoule dans la perforation. Il faut ajouter du fil de soudure jusqu'à ce que la perforation soit complètement fermée (voir fig. 3.4). Ensuite il faut retirer d'abord le fil de soudure et ensuite le fer et attendre le refroidissement de l'emplacement. Il ne faut en aucun cas bouger le composant pendant le refroidissement afin d'éviter des faux contacts par la suite.

Les pastilles qui sont connectées sur la surface cuivrée au dessus et en dessous peuvent demander un peu plus d'apport thermique jusqu'à ce que le fil de soudure se soit écoulé dans la perforation.

Les mauvaises brasures se reconnaissent par des dépôts de soudure ronds sur la pastille ou une surface mate (ou très mate pour la soudure sans plomb). Il faut les rattraper.

Pour implanter les sockets ou autres pièces qui doivent être couchées à plat sur la platine, il existe une astuce : Il faut d'abord braser un seul fil du composant. Ensuite on appuie légèrement avec les doigts sur la pièce et chauffe encore une fois la soudure (Attention : La pièce peut devenir très chaude), pour que la pièce puisse se poser sur la platine. Ensuite il faut braser les autres fils et liquéfier encore une fois la première brasure avec un peu de fil de soudure. Lorsque le composant est brasé, les extrémités des fils qui dépassent doivent être coupées à ras de la platine au moyen d'une pince coupante de côté, sans tirer sur le fil.



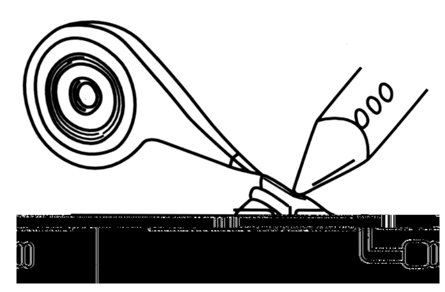
Lorsque vous coupez les fils, veillez à ne blesser personne avec les fils qui sont éjectés. Evidemment les composants sur le côté supérieur de la platine ne doivent pas se toucher. Au besoin, il faut les redresser.

3.4. Dessoudage de composants mal montés

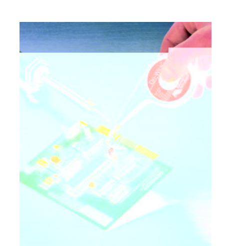
Si un composant a été mal implanté, il faut le retirer. Comme ASURO possède une platine double face avec des perforations à double contact, l'entreprise n'est pas facile.

La procédure suivante a fait ses preuves :
D'abord il faut liquéfier tous les points de brasage du composant en question (éventuellement à apportant encore un peu de fil de soudure) et retirer le composant à l'aide d'une pince. Ensuite, il faut dégager les trous à l'aide d'une tresse à dessouder.

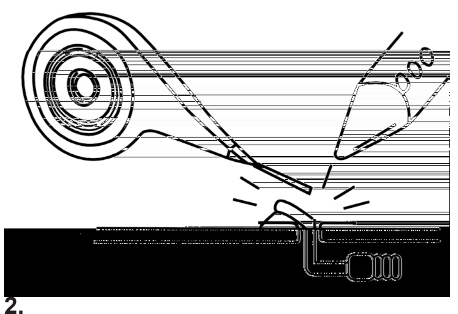
A cet effet, il faut placer la tresse à dessouder sur l'emplacement de la soudure et chauffer les deux jusqu'à ce que la tresse ait absorbé la soudure. Ensuite, il faut retirer le fer à souder et le fil de soudure. Il peut s'avérer utile d'absorber aussi le fil de soudure sur l'autre face.



1.
Poser la tresse à dessouder sur la soudure et chauffer les deux en même temps.



Tresse à dessouder professionnelle



2.
Retirer le fer à souder et la tresse dès que la tresse a absorbé la soudure.

4. Implantation

Avez-vous lu les consignes du brasage? Vraiment?

4.1. Implantation du transmetteur infrarouge RS232

• IC1: On commencer par souder uniquement le socket à 8 broches dont la polarité doit correspondre à la polarité marquée sur la platine.

• D1, D2, D3: 1N4148, respectez la bonne polarité! Ne pas confondre avec le ZPD5.1 ou BZX55-C5V1 (Impression)!

• D4: ZPD5.1 ou BZX55-C5V1, respectez la bonne polarité!
Ne pas confondre avec le 1N4148 (Impression)!

• C2, C4: 100nF céramique, Impression: 104

• C3: 680pF céramique, impression: 681

• Q1: BC547 (A,B ou C) ou BC548 (A,B ou C)

• R1, R5: 20k Ohms, 5% (rouge, noir, orange, doré)

• R2: 4.7k Ohms, 5% (jaune, violet, rouge, doré)

• R3: 470 Ohms, 5% (jaune, violet, marron, doré)

• R6: 10k Ohms, 5% (marron, noir, orange, doré)

• R7: 220 Ohms (rouge, rouge, marron, doré)

• C1: 100 µF/ au moins 16V, respectez la bonne polarité!

• TR1: condensateur 10k Ohms

• D5: SFH 415-U LED IR (boîtier noir), respectez la bonne polarité!
Le boîtier doit reposer à plat sur la platine!

• IC2: SFH5110-36 CI récepteur IR, plier les fils avec une pince!
Respectez la bonne polarité (la face bombée doit être orientée vers le dessus), Attention: Sensibilité électrostatique et –avis aux soudeurs débutants – sensible à la chaleur !

• X1: Fiche Sub-D femelle à 9 contacts, Le boîtier doit reposer à plat sur la platine.
Il faut également souder les languettes de fixation !

• IC1: Insérer le NE555P, respecter la polarité indiquée (nez ou cercle)!

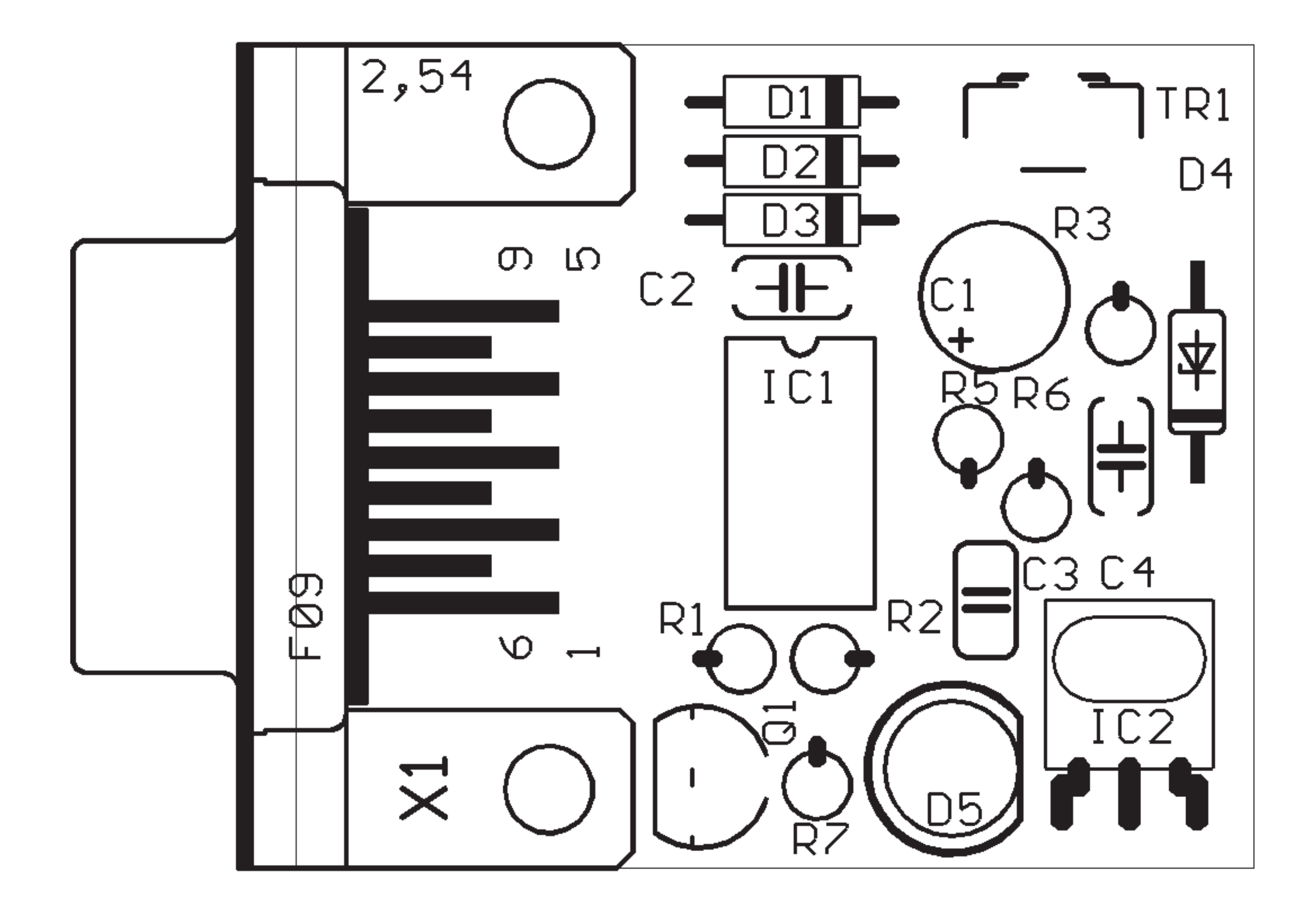


Fig. 4.1.: Implantation du transmetteur infrarouge RS232

Jetez un dernier regard critique sur les soudures et vérifiez la bonne liaison et l’absence de courts-circuits et rattrapez les défauts le cas échéant.

Fini!

4.2. Transmetteur Infrarouge USB fini

Un transmetteur IR USB fini est disponible en option.

Fig. 4.2.: Transmetteur Infrarouge USB

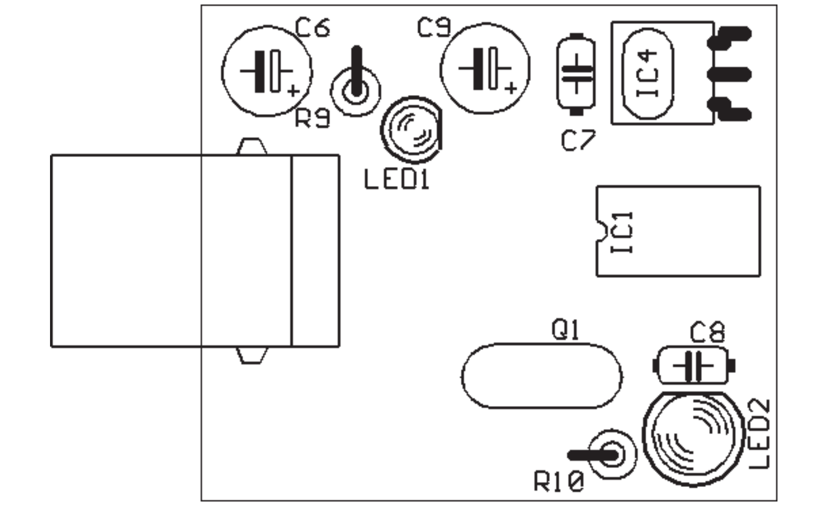


Fig. 4.3.: Vue de dessus du transmetteur Infrarouge USB

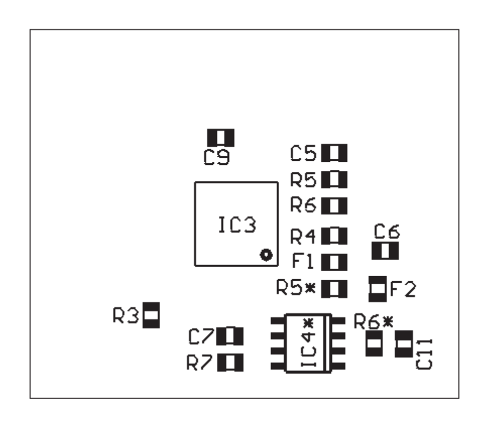


Fig. 4.4.: Vue de dessous du transmetteur Infrarouge USB

4.3. Implantation de la platine ASURO

Les deux axes plus longs pour le deuxième niveau d'engrenage sont soudés ou collés sur le dessous. La soudure est préférable car les corrections sont plus simples à effectuer et le refroidissement est plus rapide que d'attendre que la colle ait prise.

Les deux axes plus courts se placent sur le dessus et plus en direction du centre de la platine. Avant l'implantation, vous pouvez nettoyer les axes à l'endroit de la colle ou de la soudure (pas sur la surface de roulement) avec du papier de verre très fin (grain 240 ou plus) pour qu'ils absorbent mieux la colle ou le fil de soudure. Si vous décidez de souder, il est recommandé de procéder de la manière suivante :

Fixez d'abord les axes plus longs. A cet effet, posez la platine sur le dessus et placez l'axe jusqu'à la butée dans la fente. L'axe doit reposer sur toute la longueur. Etamez la panne et appuyez l'axe avec la panne sur la platine. Dès que l'axe est chaud, ajoutez du fil de soudure aux points de contact. De cette façon vous reliez l'axe à la platine. Dès que l'axe a été soudée tout autour, il faut le pousser davantage avec un tournevis sur la platine et retirer le fer à souder. Pour souder l'axe, il est conseillé d'augmenter la température du fer (env. 420°C) et d'utiliser une panne large d'env. 3mm. Pour les composants électroniques, il faut bien sûr baisser la température à env. 360°C.

Lorsque l'ensemble est refroidi, il faut souder le deuxième axe sur le dessous de la platine. Ensuite, c'est le tour des deux axes sur le dessus.
La fig. 4.5 représente la platine après le montage des axes.

Lorsque les axes sont refroidis, il faut insérer les roues dentées. Les dents des roues doivent mordre proprement et les roues doivent tourner facilement. Si ce n'est pas le cas, les axes ont été soudés de travers et doivent être redressés (ce n'est pas une punition mais un exercice) ou alors des résidus de soudure se trouvent dans la partie à l'extérieur de la platine et doivent être retirés. Le mieux est de se servir d'une lime fine ou de papier de verre.
Quand tout s'insère correctement, mettez de côté les roues dentées pour implanter tous les autres composants à leur emplacement dédié sur la platine.

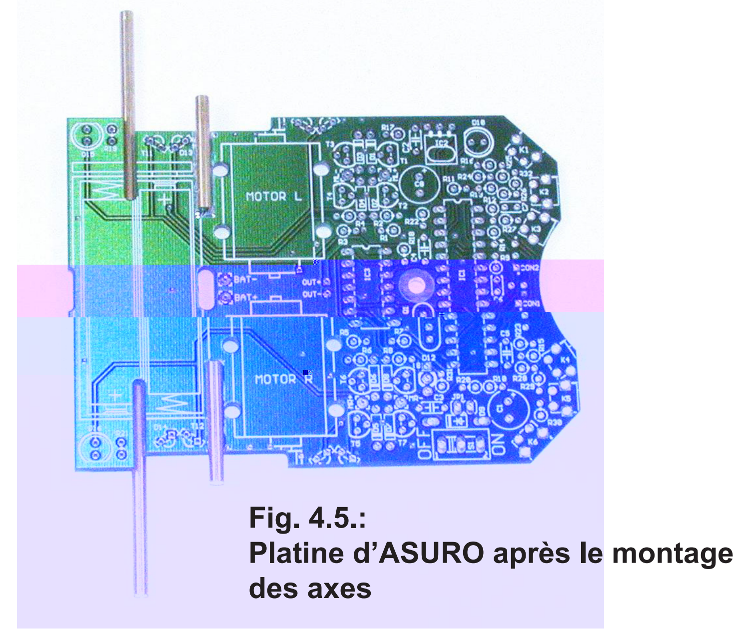
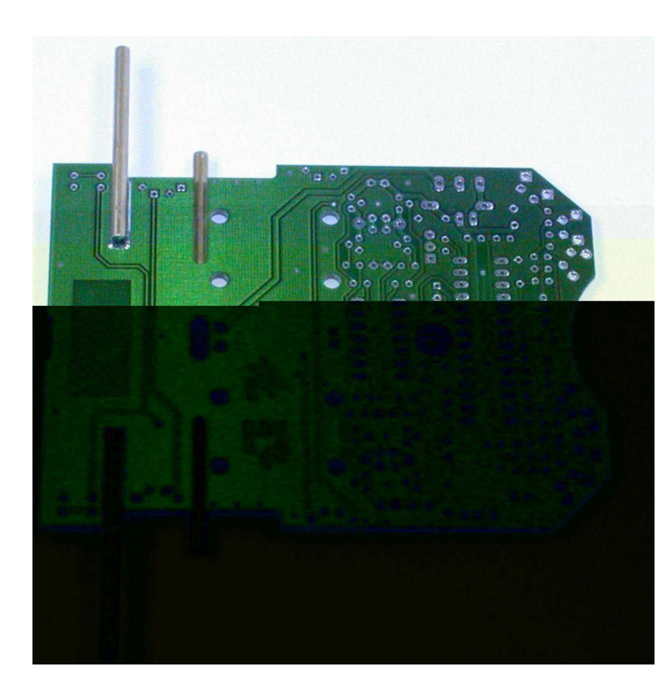


Fig. 4.5.:
Platine d'ASURO après le montage des axes



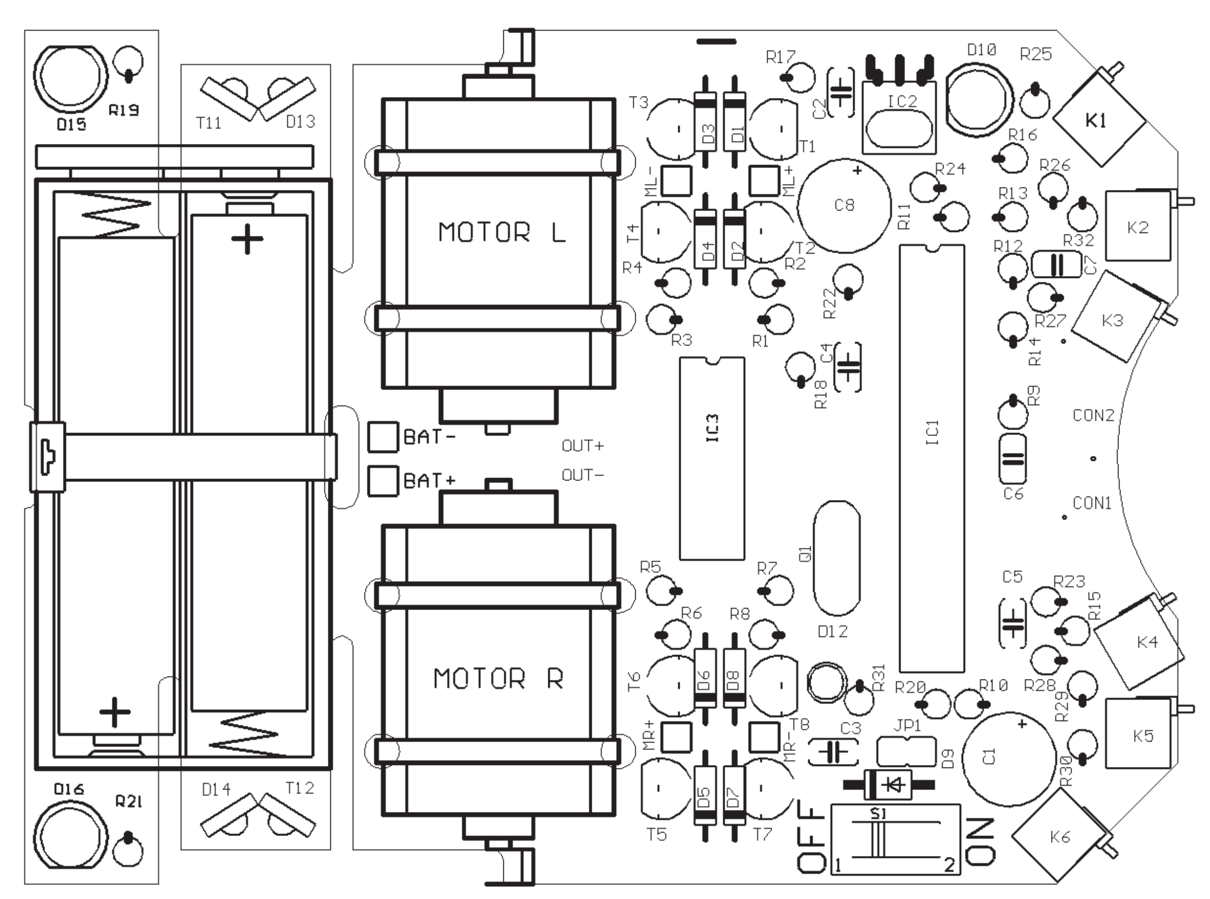


Fig. 4.6.: Implantation sur le dessus de la platine principale d'ASURO

L'implantation se fait dans l'ordre suivant:

- IC1: D'abord seulement le socket. Il faut monter soit un 28 broches (si disponible) ou deux à 14 broches l'un derrière l'autre. Respectez la bonne polarité (l'encoche est dirigée vers l'encoche de l'impression d'implantation sur la platine) !

- IC3: Seulement le socket, 14 broches. Respectez la bonne polarité (l'encoche est dirigée vers l'encoche de l'impression d'implantation sur la platine)!

- K1, K2, K3, K4, K5, K6: Capteurs. A placer le plus droit possible sur la platine !

- Q1; Oscillateur 8MHz

- D1, D2, D3, D4, D5, D6, D7, D8: 1N4148; respectez la bonne polarité!

- D9: 1N4001; respectez la bonne polarité!

- JP1: Connecteur à 2 broches. Les broches courtes sont soudées, le cavalier correspondant n'est pas encore implanté!

- D12: LED bicolore, diamètre 3mm, trois fils de connexion. Attention à la polarité. (Le repérage peut être différent. Dans tous les cas : le fil le plus court doit aller dans le tracé carré)!

- C2, C3, C4, C5: 100nF céramique; impression: 104

- C6, C7: 4,7nF céramique; impression: 472

Electronique

- T1, T3, T5, T7: BC327-40 ou BC328-40
- T2, T4, T6, T8: BC337-40 ou BC338-40
- R1, R2, R3, R4, R5, R6, R7, R8, R19, R21: 1k Ohm, 5% (marron, noir, rouge, doré)
- R9, R16: 220 Ohms, 5% (rouge, rouge, marron, doré)
- R10, R17, R22, R31: 470 Ohms, 5% (jaune, violet, marron, doré)
- R11: 100 Ohms, 5% (marron, noir, marron, doré)
- R12: 12k Ohms, 1% (marron, rouge, noir, rouge, marron)
- R13: 10k Ohms, 1% (marron, noir, noir, rouge, marron)
- R14, R15: 20k Ohms, 5% (rouge, noir, orange, doré)
- R18, R20: 4,7k Ohms, 5% (jaune, violet, rouge, doré)
- R23: 1M Ohm, 5% (marron, noir, vert, doré)
- R24: 1k Ohm, 1% (marron, noir, noir, marron, marron)
- R25, R26, R32: 2k Ohms, 1% (rouge, noir, noir, marron, marron)
- R27: 8,2k Ohms, 1% (grau, rouge, noir, marron, marron)
- R28: 16k Ohms, 1% (marron, bleu, noir, rouge, marron)
- R29: 33k Ohms, 1% (orange, orange, noir, rouge, marron)
- R30: 68k Ohms, 1% (bleu, grau, noir, rouge, marron)
- C1, C8: Elko 220 µF 10V ou plus, respectez la bonne polarité!
- IC2: SFH5110-36 CI transmetteur infrarouge, plier les fils à l'aide d'une pince! Respectez la bonne polarité (Côté bombée vers le haut).
  Attention:
  Sensibilité électrostatique et –avis aux soudeurs débutants – sensible à la chaleur !
- D10: SFH 415-U IR-LED 5mm; boîtier noir. Respectez la bonne polarité! Le boîtier doit reposer sur la platine!
- T11, T12: LPT80A, Transistor photoélectrique. Boîtier incolore doit reposer sur la platine. Respectez la bonne polarité!
- D13, D14: IRL80A, IR-LED, Boîtier rose doit reposer sur la platine. Respectez la bonne polarité!
- D15, D16: LED 5mm rouge, boîtier rouge, respectez la bonne polarité
  (fil court sur la face repérée)!
- S1: Interrupteur Marche/Arrêt

Electronique

Il manque encore trois composants (ils permettent de suivre un tracé). Il faut cependant les implanter sur le dessous de la platine et les souder par le dessus voir fig.4.7.):

- T9, T10: SFH300, Transistor photoélectrique 5mm, respectez la bonne polarité!
  Ils sont légèrement écartés de la platine.

- D11: LED 5mm rouge, boîtier rouge, respectez la bonne polarité (fil court sur la face repérée)!

Fig. 4.8. présente la platine implantée jusqu'à présent par le dessus et le dessous

Voilà, c'était tout ce qu'il fallait en matière de composants électroniques. Maintenant il faut monter les composants électromécaniques et mécaniques.

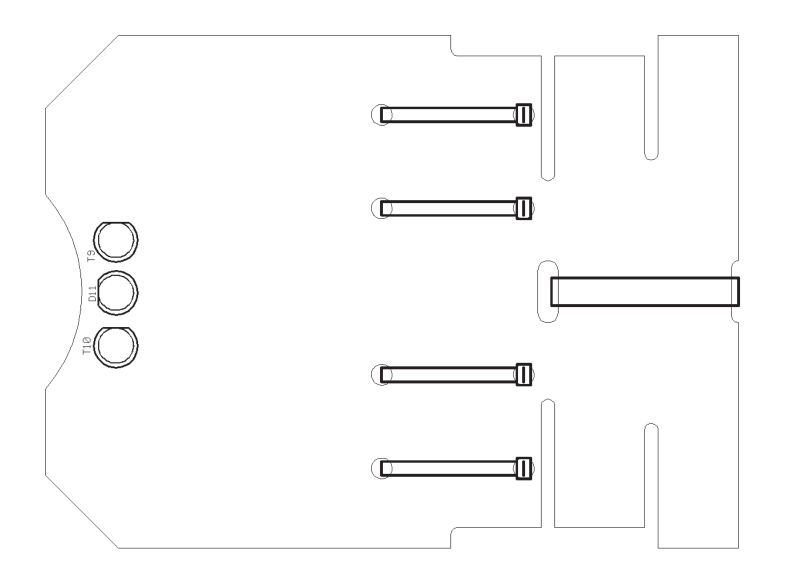
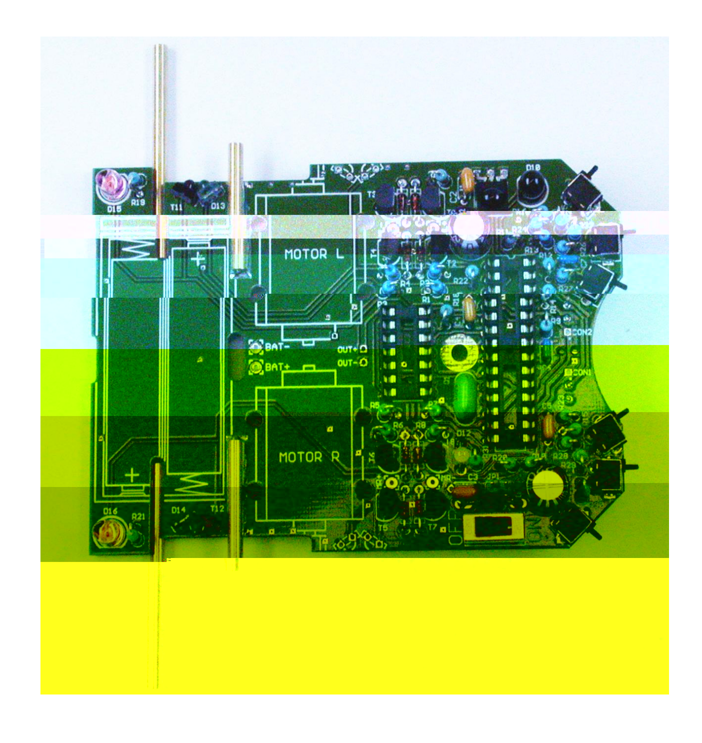


Fig. 4.7.:
Implantation de la face inférieure de la platine d'ASURO



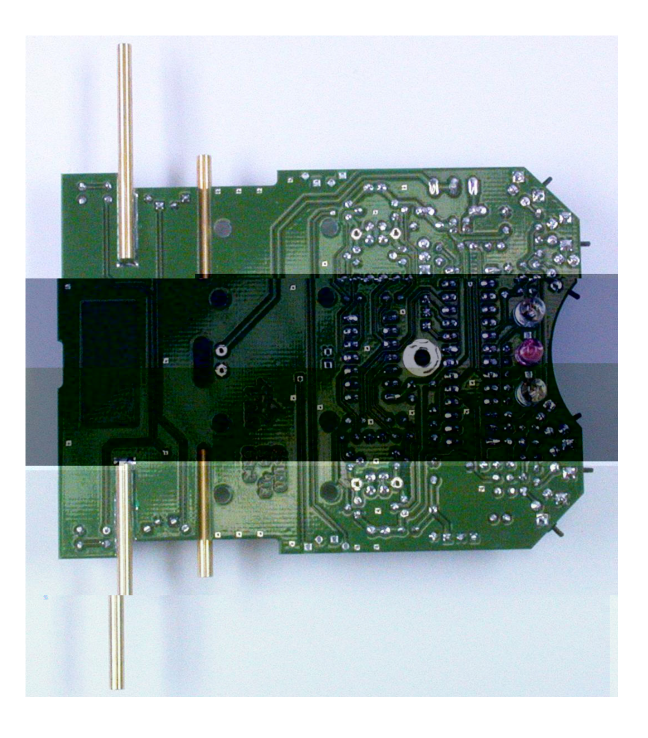


Fig. 4.8.: ASURO implanté par le dessus et le dessous

4.4. Montage du moteur

Lorsque l’implantation de la platine d’ASURO est terminée, il n’y a plus qu’à monter les câbles sur les moteurs et les fixer provisoirement.
Pour la connexion du moteur, il vous faut un câble noir et un rouge d’env. 70mm de longueur avec des extrémités dénudées et étamées. Si les câbles fournis n’ont pas encore été préparés, vous devez dénuder le câble aux extrémités sur une longueur d’env. 4mm, le tourner et l’étamer en le mettant en contact avec le fer à souder et un peu de fil de soudure. Vous pouvez ensuite retirer les résidus gênants de soudure au moyen d’une pince coupante de côté.
Soudez maintenant le câble rouge sur le côté du moteur marqué d’un point rouge ou d’un signe « Plus » et le câble noir sur l’autre.

Les câbles de branchement de chaque moteur sont ensuite entrelacés (ce n’est pas indispensable mais apporte des avantages en matière de compatibilité électromagnétique et est d’un plus bel aspect…).

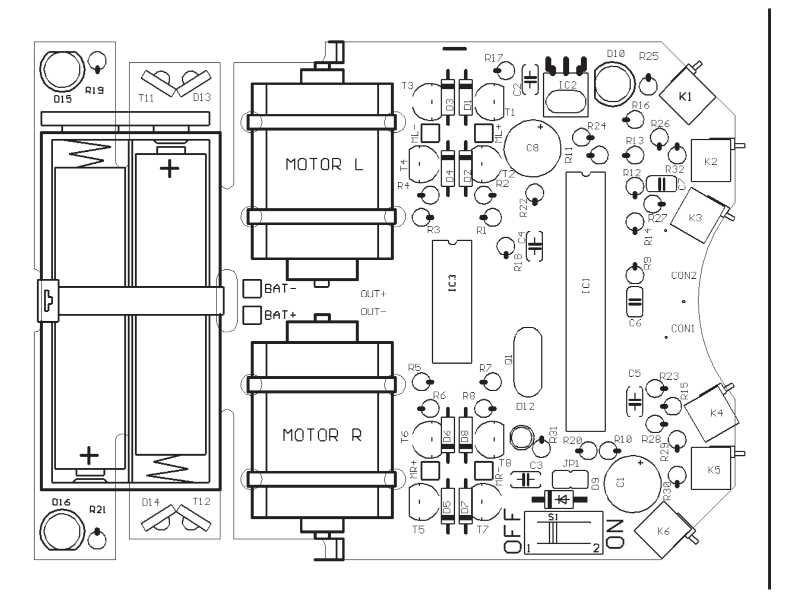
Le câble rouge du moteur gauche est soudé dans „ML+“ et le noir dans „ML-„; le câble rouge du moteur droit est soudé dans « MR+ » et le noir dans « MR-« .
Maintenant il faut installer provisoirement les moteurs sur la platine. A cet effet, il faut passer le collier de câble dans les trous de la platine à côté des moteurs de façon à ce que les têtes des colliers restent en dessous de la platine et passer le collier autour des moteurs.

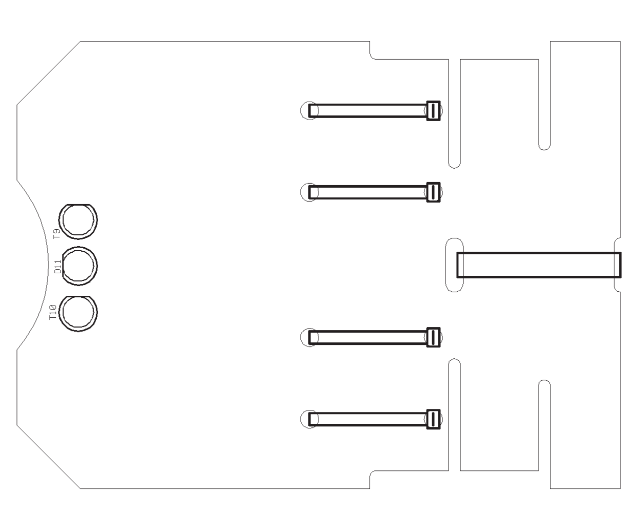
4.5. Alimentation électrique



Si ASURO doit être alimenté par piles, le cavalier JP1 doit impérativement être ouvert ! Si vous utilisez des batteries, il faut le fermer. Une mauvaise polarité des batteries lorsque le cavalier est fermé détruit l’électronique !

Le câble rouge du support de piles est soudé (sans les piles) dans BAT+ et le câble noir dans BAT-. Vérifiez ensuite que l’interrupteur est en position OFF et que les 4 piles ou batteries sont insérées avec la bonne polarité dans le support. Il faut fixer le support de piles tout de suite ou après la mise en service en passant le plus long collier de câbles (amovible) par le trou dans la platine.





5. Mise en service et Test

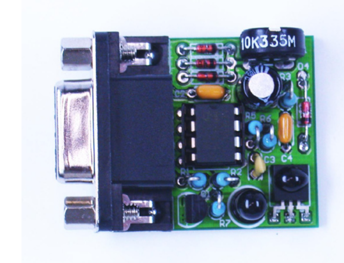
Enfin tout est monté et l'aventure peut commencer. Cependant, il faut chercher, trouver et éliminer les erreurs qui se sont glissées et cela sans faire trop de dégâts.

5.1. Transmetteur Infrarouge RS232

Cette mise en service ne s'applique qu'au transmetteur RS232 à infrarouge.
Tout d'abord il faut vérifier le bon fonctionnement du transmetteur IR RS232 puisque nous en aurons besoin par la suite pour l'auto-test du véhicule. A cet effet, le transmetteur RS232 sera branché sur une interface série libre au moyen de la rallonge sérielle fournie.

Ensuite, il faut démarrer le programme Terminal de Windows « Hyper Terminal » (sous Linux p.ex. « Minicom »). Normalement, vous le trouverez sous Démarrer --> Tous les Programmes --> Accessoires --> Communications --> Hyper Terminal. S'il ne s'y trouve pas, vous devez l'installer à partir du CD Windows.

Les programmes de terminal datent en fait encore de l'époque des modems et avant lorsque les communications avec d'autres ordinateurs se faisaient fréquemment par le biais de l'interface série. Aujourd'hui ils sont surtout utilisés pour se logger en base texte sur un autre ordinateur par Internet. Après le démarrage de Hyper Terminal, le programme demande un nom pour la nouvelle connexion. Ici, vous pouvez saisir ASURO et sélectionner un symbole de votre choix. Dans la fenêtre suivante, vous devez sélectionner sous « Se connecter en utilisant : » l'interface COM sur laquelle vous avez branché le transmetteur. Après avoir cliqué sur OK, vous sélectionnez



• Bits par seconde: 2400
• Bits de données: 8
• Parité: Aucune
• Bits d'arrêt: 1
• Contrôle de flux: Aucun
Confirmer avec “OK”.

Tenez maintenant le transmetteur IR à env. 10cm au dessus d'une feuille de papier blanche. Les composants sont face au papier. Tapez maintenant quelques touches sur le clavier. Le programme Terminal devrait afficher ces touches. Le transmetteur IR envoie la pression sur la touche par le biais de la diode IR (D5). Le signal réfléchi par le papier est transmis au CI récepteur (IC2) et de là au PC. Si aucun caractère ou des caractères erronés s'affichent, vous pouvez tourner doucement avec un petit tournevis le trimmer entre sa butée gauche et droite et appuyer à nouveau sur quelques touches, jusqu'à ce que les bons caractères s'affichent.

L'ensemble ne fonctionne pas comme décrit ? Dommage. Cela signifie qu'il y a une erreur qu'il faut corriger (voir chapitre 6.1).
Pour être tout à fait sûr, vous pouvez débrancher le transmetteur IR et appuyer à nouveau sur quelques touches. Aucun caractère ne doit s'afficher.

5.2. Transmetteur Infrarouge USB

Cette mise en service ne s'applique qu'au transmetteur Infrarouge USB.
Attention ! Le transmetteur USB n'est pas protégé par un boîtier et sensible aux décharges électrostatiques. Avant l'utilisation, vous devez toucher un corps métallique (radiateur, boîtier d'ordinateur) pour décharger l'électricité statique éventuelle afin d'éviter tout dommage. Vous avez aussi la possibilité d'installer le transmetteur dans un boîtier qui laisse passer la lumière infrarouge.

5.2.1 Windows

Le transmetteur USB est branché sur un port USB lire au moyen du cordon USB. Le message s'affiche:

"Un nouveau matériel a été détecté: AREXX ASURO USB-IR-Transceiver"
Installez ensuite le driver USB à partir du CD d'ASURO. Si l'ordinateur ne trouve pas automatiquement le driver, sélectionnez le répertoire (D : représente ici le lecteur CD-ROM)

D:\Windows\USB_Driver" . Vous aurez éventuellement besoin des droits d'administrateur. Dans ce cas, fermez la session et ouvrez une nouvelle session en tant qu'administrateur. Maintenant un driver est installé qui permet d'adresser le transmetteur USB sous Windows comme une interface série normale.

Si tout cela s'est passé sans problème, vous pouvez démarrer le programme „Hyper Terminal", en indiquant comme nom de connexion ASUROUSB et en choisissant un symbole. Dans la fenêtre suivante « Se connecter en utilisant : », vous sélectionnez la dernière interface COM libre. Appuyez sur « OK » et sélectionnez :

- Bits par seconde: 2400
- Bits de données: 8
- Parité: Aucune
- Bits d'arrêt: 1
- Contrôle de flux: Aucun

Confirmer avec "OK".



Tenez maintenant le transmetteur avec la face de la diode luminescente vers le bas à 10cm au-dessus d'une feuille de papier blanche. Si vous utilisez le transmetteur sans boîtier, vous ne devez tenir la platine que par la fiche ou au bord afin de ne pas perturber le circuit. Tapez maintenant sur quelques touches dans le programme Terminal. La LED jaune sur la platine doit clignoter et les touches tapées doivent s'afficher sur l'écran.

Si cela ne fonctionne pas, continuez à lire à partir du chapitre 6.2.

Si tout a bien fonctionné, vous pouvez continuer avec la mise en service de la platine ASURO.

5.2.2 Linux

Le transmetteur USB est branché sur un port USB libre au moyen du cordon USB. Vous entendrez un petit ‘Bip’ lorsque Linux a reconnu le transmetteur. Afin de vérifier si l’appareil a été correctement identifié, vous pouvez consulter l’inscription correspondante dans le répertoire proc:

foo@bar:/>cat /proc/tty/driver/usb-serial
Ce qui doit générer une édition qui comporte au moins les inscriptions suivantes (au lieu du “0:” Cela peut aussi être “1:”, “2:” etc.):
usbserinfo:1.0 driver:v1.4
0: module:ftdi_sio name:”FTDI 8U232AM Compatible” vendor:0403 product:6001
num_ports:1 port:1 path:usb-00:11.2-1

Pour effectuer un test, vous devez configurer Minicom sur l’interface /dev/ttyUSB0 (ou 1, 2 etc...) et les paramètres suivants:

• Bits par seconde: 2400
• Bits de données: 8
• Parité: Aucune
• Bits d’arrêt: 1
• Contrôle de flux: Aucun

Confirmer avec “OK”.

Pour cela, il faut éventuellement avoir les droits de source.
Il peut également être nécessaire d’accorder des droits de lecture et d’écriture à l’utilisateur ou au groupe d’utilisateurs souhaités sur le Device /dev/ttyUSB?. Ceci peut se faire avec un chmod u+rw /dev/ ttyUSB0 (ou 1, 2...) ou chmod g+rw /dev/ttyUSB0 (également avec des droits de source).

Tenez maintenant le transmetteur avec la face de la diode luminescente vers le bas à 10cm au-dessus d’une feuille de papier blanche. Si vous utilisez le transmetteur sans boîtier, vous ne devez tenir la platine que par la fiche ou au bord afin de ne pas perturber le circuit.

Tapez maintenant sur quelques touches dans le programme Terminal. La LED jaune sur la platine doit clignoter et les touches tapées doivent s’afficher sur l’écran.

Si cela ne fonctionne pas, continuez à lire à partir du chapitre 6.2.

Si tout a bien fonctionné, vous pouvez continuer avec la mise en service de la platine ASURO.

5.3. Mise en service de la platine ASURO



Le processeur (IC1) n’a pas encore été installé à ce stade !

Maintenant croisez les doigts et mettez l’interrupteur sur ON. Les deux LED (D15, D16) devraient luire faiblement. Si ce n’est pas le cas, mettez immédiatement l’interrupteur principal sur OFF et continuez à lire à partir du chapitre 6.3. Tout va bien ? Alors, mettez l’interrupteur sur OFF et implantez l’IC1 (processeur) et l’IC3 (composant AND) (voir Fig. 5.1).

Eventuellement il faut plier un peu les contacts des CI pour que tous les contacts puissent s’insérer dans la bonne position dans le socle. Le mieux est de prendre le CI sur le côté et appuyer les contacts légèrement contre le bord d’une table.



Le processeur IC1 ATmega8, le composant IC3 CD4081 et le récepteur IR IC2 SFH5110-36 sont sensibles à l’électricité statique.
Cela signifie qu’ils peuvent être détruits par un simple contact si la personne est chargée électriquement ce qui peut arriver en traversant simplement un tapis.
Avant de manipuler ces composants, il est conseillé de se relier à la terre au moyen d’un ruban de masse ou tout du moins toucher le boîtier métallique d’un appareil ou un radiateur.

Fig. 5.1.: ASURO après l’implantation des CI

Implanter le straps (J1) uniquement en cas d'alimentation par batterie. Les fentes des IC doivent correspondre à la fente du socle s'il a été correctement implanté. Le processeur a déjà été programmé d'usine avec un auto-test et vérifiera tous les composants après la mise sous tension.

Afin d'éviter les problèmes, lisez d'abord complètement le chapitre suivant avant la mise sous tension et revenez ensuite sur ce passage.

Allons-y ! Mettez l'interrupteur sur ON et ne perdez pas ASURO de vue.



Si le programme Hyper Terminal (Windows) ou Minicom (Linux) tourne, le transmetteur IR est branché et ASURO est sous étroite observation, vous pouvez suivre l'auto-test sur l'écran.

5.3.1. Eléments visuels

La LED d'état (D12) s'allume brièvement en orange et les « LEDs arrières» (D15, D16) luisent également mais très faiblement. Si ce n'est pas le cas, mettez l'interrupteur immédiatement sur OFF (voir chapitre 6.3.3). Ceci était la phase Boot d'ASURO. Maintenant tous les éléments visuels sont vérifiés individuellement pendant env. 3 secondes dans l'ordre suivant:

• LED d'état (D12) verte
• LED d'état (D12) rouge
• LED avant (D11) sur le dessous d'ASURO
• LED arrière (D15) gauche
• LED arrière (D16) droit
• Tous les éléments visuels en même temps

Si, contre toute attente, une erreur se produit, il faut immédiatement éteindre ASURO et corriger l'erreur (voir chapitre 6.4), puisque tous les éléments visuels qui viennent d'être testés, sont indispensables pour les autres tests.

5.3.2. Phototransistors (T9, T10)

Après le test des éléments visuels, la LED d'état (D12) doit s'allumer en vert. Ceci signifie que les phototransistors sont maintenant vérifiés (env. 10 secondes) qui se trouvent sur le dessous d'ASURO et qui sont nécessaires pour suivre un tracé. Lorsque les phototransistors (T9, T10) sont éclairés, la LED arrière (D15, D16) correspondante doit s'allumer et s'éteindre lorsque le transistor n'est plus éclairé. Le transistor droit (T10) dépend de la LED arrière droite (D16) et le transistor gauche (T9) de la LED arrière gauche (D15). Il est possible que la LED arrière continue à luire faiblement à l'état éteint. Ceci est normal. En cas d'erreur, l'auto-test peut se poursuivre et l'erreur peut être corrigée ultérieurement.

5.3.3. Commutateurs

ASURO est arrêté, tous les éléments visuels sont éteints. C'est bon signe ! Maintenant les commutateurs sont vérifiés (env. 15 secondes). Appuyez un peu partout et observez ce qui se passe.

L'affectation se présente de la manière suivante:
K1 --> LED d'état (D12) s'allume en vert
K2 --> LED d'état (D12) s'allume en rouge
K3 --> LED avant (D11) en dessous d'ASURO
K4 --> LED arrière gauche (D15)
K5 --> LED arrière droite (D16)
K6 --> Le moteur gauche tourne (si le moteur ne tourne pas, vous pouvez néanmoins poursuivre l'auto-test. Les entraînements sont testés séparément. Une erreur éventuelle dans l'entraînement du moteur (voir chapitre 6.8) peut être corrigée à ce moment-là.
L'activation de plusieurs détecteurs engendre une combinaison des signaux. En cas d'erreur, l'auto-test peut se poursuivre. L'erreur pourra être corrigée ultérieurement.

5.3.4. Détecteur photoélectrique (Odométrie)

La LED de tracé (D11) en dessous d'ASURO s'allume pour indiquer le début du test suivant (env. 15 secondes) des détecteurs photoélectriques pour l'odométrie. La LED d'état s'allume lorsqu'un papier blanc est placé devant le détecteur. Si le papier est tenu devant le détecteur gauche (T11), la LED d'état (D12) s'allume en vert. Si vous tenez un papier devant le détecteur droit (T12), la LED (D12) doit s'allumer en rouge. Lorsque vous retirez le papier, la LED s'éteint. Cela signifie qu'un passage clair/sombre est détecté et l'odométrie fonctionne. En cas d'erreur, l'auto-test peut se poursuivre et l'erreur peut être corrigée ultérieurement.

5.3.5. Entraînements

Les deux LEDs arrières (D15, D16) s'allument clairement. L'avant-dernier test commence (env. 15 secondes). Les entraînements sont vérifiés. Le moteur gauche est poussé en marche avant en départ arrêté jusqu'au nombre maximal de tours et ensuite le régime diminue jusqu'à l'arrêt. Le sens de rotation change et poussé à nouveau de l'arrêt jusqu'au nombre de tours maximum. Le moteur droit doit subir la même procédure. Ensuite les deux moteurs sont entraînés en même temps. Et voilà pour la dernière fois la fameuse remarque : En cas d'erreur, l'auto-test peut se poursuivre et l'erreur peut être corrigée ultérieurement.

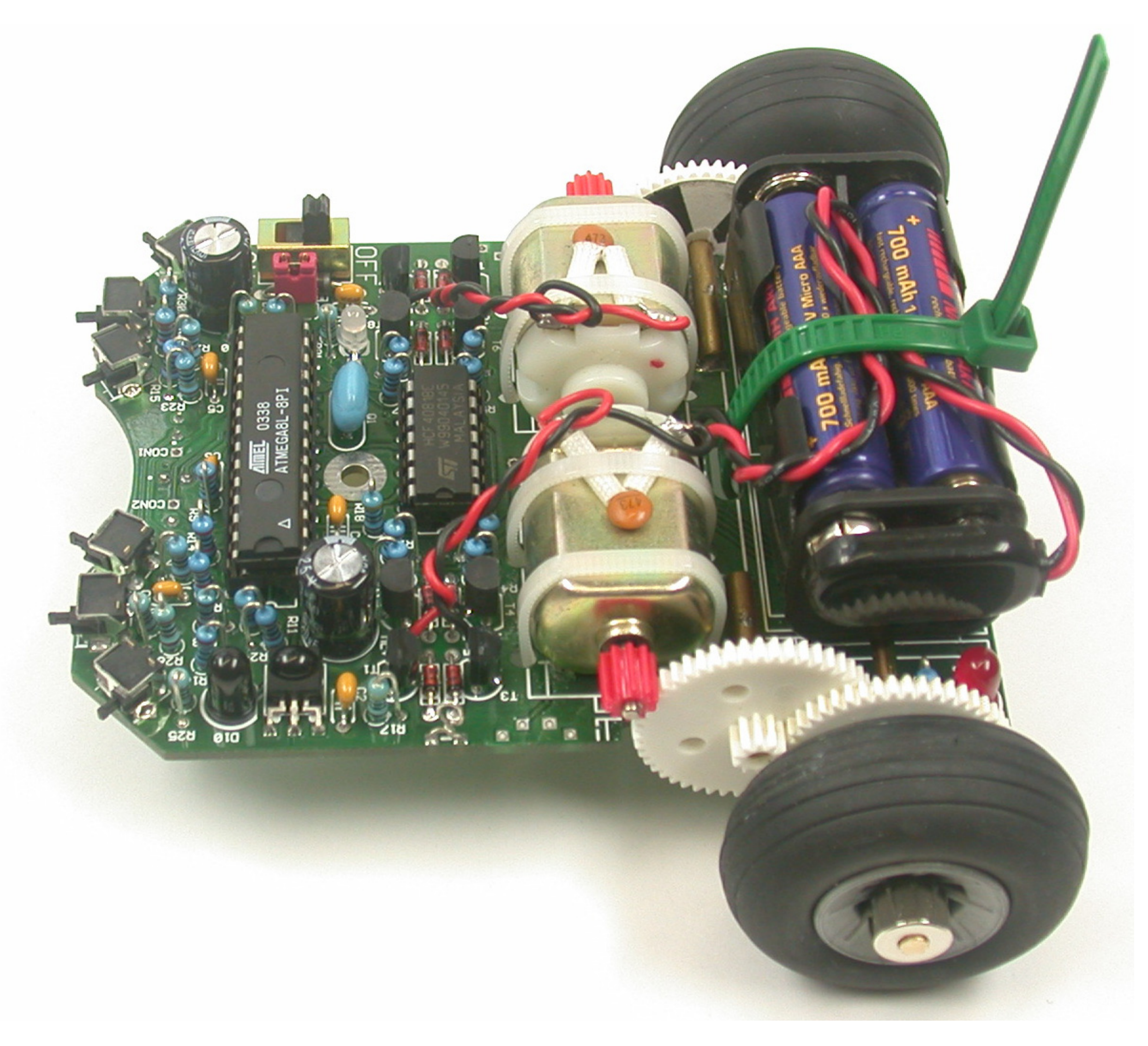
5.3.6. Transmetteur IR

Lorsque la LED d'état clignote en jaune, la dernière partie (env. 15 secondes) bat son plein.
Le transmetteur IR envoie ou reçoit des données. Pour pouvoir les recevoir, il faut brancher le transmetteur IR assemblé sur le PC et utiliser un programme de terminal comme le « Hyper Terminal » de Windows. La configuration est la même que pour le test du transmetteur IR. ASURO répond aux signes reçus par les signes suivants dans l'alphabet. Si les données n'arrivent pas dans un temps imparti, ASURO envoie un 'T'. A chaque signe envoyé, la LED rouge s'ajoute à la LED verte ce qui explique le clignotement jaune.

Si un contact visuel est établi entre ASURO et le transmetteur infrarouge (env. 50cm de distance), un ‘T’ s’affichera régulièrement dans le programme ‘Terminal’ ou bien la touche appuyée sur le PC apparaît une fois (un signal envoyé et réfléchi par le transmetteur) suivie par la lettre suivante de l’alphabet. P.ex.:
La touche « e » est appuyée => Le programme Terminal affiche “ef”
La touche « j » est appuyée => Le programme Terminal affiche “jk”
La touche « 3 » est appuyée => Le programme Terminal affiche “34”
En cas d’erreur reportez-vous au chapitre 6.9.

5.3.7. Fini ?

Si une erreur s’est produite, il faut couper la batterie et corriger l’erreur au moyen du chapitre 6 dans la mesure où aucun des composants n’a été endommagé. Ensuite il faut répéter l’auto-test. Après cela, ASURO est réellement prêt à l’emploi et il est certain que dans l’avenir, une erreur proviendrait du logiciel et non pas du matériel. Si vous effectuez un auto-test plus tard après avoir exécuté plusieurs programmes sur ASURO, vous devez programmer ASURO avec le fichier « SelfTest.hex » à partir du CD.



6. Diagnostic de défaillance

6.1. Le transmetteur IR RS-232 ne fonctionne pas !

6.1.1. La touche et le signe affiché ne concordent pas

Tourner le trimmer TR1 jusqu'a ce que la touche frappée et le signe affiché, concordent.

6.1.2. Le programme Terminal n'édite pas de signes

Est-ce que le circuit Programmateur (IC1) est monté à l'endroit (l'encoche est dirigée vers les 3 diodes) ? Prendre la télécommande infrarouge d'un appareil quelconque (magnétoscope, téléviseur, tuner, etc.) et la diriger sur le transmetteur IR et appuyer sur quelques touches. Si rien ne se passe, vérifier à nouveau les réglages du port COM. Si le programme Terminal affiche des signes erratiques, la partie réception fonctionne (IC2, R3, C4, D4, T1). Toutes les autres parties doivent être vérifiées.

6.1.3. Cela ne fonctionne toujours pas

Vérifier le sens de l'implantation (polarité) et la valeur de tous les composants (voir fig. 4.1). Vérifier l'absence de courts-circuits sur les brasures et la présence de soudures sèches. Est-ce qu'un point de soudure s'est décollé quelque part ? Si vous n'avez trouvé aucune erreur, vous devez chercher le composant défectueux au moyen du schéma de connexions (voir chapitre B) et d'un appareil de mesure approprié (multimètre ou oscilloscope). (IC1, IC2, Q1, D4 sont très probablement à l'origine de la défaillance)

6.2. Le transmetteur infrarouge USB ne fonctionne pas

6.2.1 Windows

Est-ce que le driver a été correctement installé ? Parfois d'autres numéros de ports COM sont attribués à la place de la dernière. Dans ce cas, vous pouvez sélectionner un autre port dans le programme Hyperterminal et répéter le test. Le cas échéant, vous pouvez également vérifier dans le système de gestion quel port a été attribué.

6.2.2 Linux

Débranchez le transmetteur USB, attendez quelques instants et rebranchez-le. Si après cela, l'inscription dans le répertoire proc n'apparaît toujours pas, il peut être utile d'installer un nouveau noyau de système d'exploitation.

6.3. Les LED arrières (D15, D16) ne luisent pas après la mise sous tension!

6.3.1. Aucune des deux LED ne luit

Observez bien et obscurcissez la pièce. Vous ne voyez toujours rien ? Alors vérifiez les points suivants:

- Est-ce que les 4 piles sont en place et chargées ou est-ce que la batterie est chargée?
- Est-ce que le câble de la batterie a été soudé avec la bonne polarité ? (rouge Bat+ / noir Bat-)
- Est-ce que la diode D9 (1N4001) a été implantée avec la bonne polarité ?
- Est-ce que R22 présente la bonne valeur? 470Ω (jaune, violet, marron, orange)
- Vérifiez éventuellement aussi:
    - R18 4,7KΩ Ω (jaune, violet, rouge, orange)
    - R19 1KΩ (marron, noir, rouge, orange)
    - R20 4,7KΩ Ω (jaune, violet rouge, orange)
    - R21 1KΩ (marron, noir, rouge, orange)

6.3.2. Seulement 1 LED sur les 2 s’allume

Est-ce que les diodes (boîtier rose) D13 (gauche), D14 (droite) ainsi que les phototransistors (boîtier transparent) T11 (gauche), T12 (droite) ont été implantés au bon endroit (voir fig. 4.3) et avec la bonne polarité ?

Est-ce que les résistances suivantes présentent la bonne valeur R18, R19 (gauche) et R20 R21 (droite) ?
4,7KΩ (jaune, violet, rouge, orange)
1KΩ (marron, noir, rouge, orange)

Est-ce que les composants sont implantés au bon endroit ? (Regardez l’impression sur la platine pour les résistances !)

6.3.3. La LED d’état (D12) ne s’allume pas en 2 couleurs après la mise sous tension

La LED d’état ne s’allume pas du tout => Voir 6.4!

La LED d’état vacille => Tension de batterie trop faible => Changer les piles/batteries. Si elles sont neuves/chargées, vérifiez les résistances R12 et R13.
12KΩ (marron, rouge, noir, rouge, marron)
10KΩ (marron, noir, noir, rouge, marron)

6.4. Un élément d’affichage ne fonctionne pas

Est-ce que le processeur a été correctement implanté ? (polarité !)

6.4.1. La LED d’état D12 ne fonctionne pas

Vérifier la polarité de la LED D12.
Vérifier la résistance R10, R31.
470Ω ((((99(jaune, violet, marron, orange)

Un test simple constitue à retirer le processeur (IC1) et d’établir le contact entre broche 7* (VCC) et broche 14 (LED d’état s’allume en vert) ou broche 4 (LED d’état s’allume en rouge). Si ce test réussit, le défaut se trouve dans le processeur ou oscillateur ou une piste est interrompue.

***** Broche 1 se trouve en haut à gauche. Ensuite on compte de la gauche vers le bas et de la droite vers le haut***

6.4.2. La LED avant D11 ne fonctionne pas

Vérifier la polarité de D11.

Vérifier la résistance R9.
220Ω (rouge, rouge, marron, orange)

Un test simple constitue à retirer le processeur (IC1) et d’établir le contact entre broche 7* (VCC) et broche 12 (LED avant s’allume en rouge). Si ce test réussit, le défaut se trouve dans le processeur ou oscillateur ou une piste est interrompue.

***** Broche 1 se trouve en haut à gauche. Ensuite on compte de la gauche vers le bas et de la droite vers le haut***

6.4.3. La LED arrière gauche D15 ne fonctionne pas

Vérifier la polarité de D15.

Vérifier les résistances R19, R18.
1kΩ (marron, noir, orange, orange)
4,7 kΩ (jaune, violet, rouge, orange)

Un test simple constitue à retirer le processeur (IC1) et d'établir le contact entre broche 7 (VCC) et broche 24 (LED arrière gauche s'allume en rouge). Si ce test réussit, le défaut se trouve dans le processeur ou oscillateur ou une piste est interrompue.

6.4.4. La LED arrière droite D16 ne fonctionne pas

Vérifier la polarité de D16.

Vérifier les résistances R21, R20.
1kΩ (marron, noir, orange, orange)
4,7 kΩ (jaune, violet, rouge, orange)

Un test simple constitue à retirer le processeur (IC1) et d'établir le contact entre broche 7 (VCC) et broche 23 (LED arrière droite s'allume en rouge). Si ce test réussit, le défaut se trouve dans le processeur ou oscillateur ou une piste est interrompue.

6.5. Le Détecteur odométrique (T9, T10) ne réagit pas

Vérifier la polarité T9, T10.

Vérifier les résistances R14 20KΩ (rouge,noir,orange,or) , R15 20KΩ (rouge,noir,orange,or).
Vérifier également si R15 n'a pas été implanté à l'envers avec R23 ou R28!

Avec un multimètre vous pouvez mesurer le signal du détecteur sur les broches 25 ou 26 lorsque le processeur a été retiré. (Obscur / 0V, clair / VCC).

6.6. Un commutateur ne fonctionne pas correctement

6.6.1. Plusieurs commutateurs ont été actionnés

Vérifier R12 12KΩ (marron,rouge,noir,rouge,marron) et R13 10KΩ (marron,noir,noir,rouge,marron).
Il faut vérifier : R25, R26, R27, R28, R29, R30, R32!

R24 1KΩ (marron,noir,noir,marron,marron)
R25 2KΩ (rouge,noir,noir,marron,marron)
R26 2KΩ (rouge,noir,noir,marron,marron)
R27 8,2KΩ (gra, rouge,noir,marron,marron)
R28 16KΩ (marron,bleu,noir,rouge,marron)
R29 33KΩ (orange,orange,noir,rouge,marron)
R30 68KΩ (bleu,gr,noir,rouge,marron)
R32 2KΩ (rouge,noir,noir,marron,marron)

Dans certains cas isolés, il peut arriver qu'en raison des tolérances des composants, l'affichage n'est pas reconnu mais ceci peut être corrigé ultérieurement par le logiciel.

6.6.2. Le comportement de l'affichage indique une inversion des commutateurs

Les résistances des commutateurs en question ont été inversées.

Il faut vérifier :
R24 1KΩ (marron,noir,noir,marron,marron)
R25 2KΩ (rouge,noir,noir,marron,marron)
R26 2KΩ (rouge,noir,noir,marron,marron)
R27 8,2KΩ (gris, rouge,noir,marron,marron)
R28 16KΩ (marron,bleu,noir,rouge,marron)
R29 33KΩ (orange,orange,noir,rouge,marron)
R30 68KΩ (bleu,gris,”

6.8. Un entraînement ne fonctionne pas

6.8.1. Aucun entraînement ne réagit

Vérifier la polarité et l'implantation d'IC3.

6.8.2. Le moteur gauche ne tourne pas du tout ou seulement dans un sens

Il faut contrôler tout le pont moteur constitué des transistors T1, T2, T3, T4 (est-ce que les bons transistors ont été implantés aux bons endroits), des diodes D1, D2, D3, D4 (polarité !) et des résistances R1, R2, R3, R4.

T1, T3 (BC327-40 ou BC328-40), T2, T4 (BC337-40 ou BC338-40)

R1, R2, R3, R4 1KΩ (marron,noir,rouge,or)

6.8.3. Le moteur droit ne tourne pas du tout ou seulement dans un sens

Il faut contrôler tout le pont moteur constitué des transistors T5, T6, T7, T8 (est-ce que les transistors ont été implantés aux bons endroits), des diodes D5, D6, D7, D8 (polarité !) et des résistances R5, R6, R7, R8.

T5, T7 (BC327-40 ou BC328-40), T6, T8 (BC337-40 ou BC338-40)

R5, R6, R7, R8 1KΩ (marron,noir,rouge,or)

6.8.4. Un moteur tourne dans le mauvais sens

Il faut intervertir les deux câbles de connexion sur le moteur qui tourne dans le mauvais sens.

6.9. Interface IR

6.9.1. ASURO n'émet pas de signaux

Vérifier la polarité de la diode IR D10.
Est-ce que la résistance R16 est correcte ?
220 Ω (rouge,rouge,marron,or)

6.9.2. ASURO ne reçoit pas de signaux

Un contact visuel doit exister entre le transmetteur IR et ASURO (distance env. 50cm) et le transmetteur doit être en état de fonctionnement (voir chapitre 6.1).

Est-ce que IC2 est correctement implanté?
Vérifier la résistance R17 et C2!
470 Ω (marron,noir,marron,or)
100nF (impression 104)

Si vous n'avez toujours pas identifié l'erreur, réfléchissez si vous n'avez pas « soudé » l'IC2 au lieu de le braser. L'IC2 est sensible à la chaleur et a pu être endommagé pendant l'implantation. Dans ce cas, vous devez vous procurer et implanter un nouveau CI (SFH 5110-36).

6.9.3. Cela ne fonctionne toujours pas parfaitement

Vérifiez la polarité de C8.
220 F/ au moins 10V

Si la transmission de données du PC vers ASURO pose toujours des problèmes, vous devez tourner un peu le trimmer TR1 du transmetteur.





La colle à deux composants a la réputation d'avoir un effet sensibilisant ce qui signifie : on développe des allergies contre de plus en plus de matières. Il ne faut donc à aucun prix qu'elle entre en contact avec la peau. Utilisez des gants en vinyle. En cas de contact, lavez vos mains immédiatement et soigneusement avec du savon ! A l'origine, la colle rapide a été développée pour la chirurgie. On s'en aperçoit lorsque deux bouts de peau collent ensemble en quelques secondes. Si cela se produit sur les doigts, vous pouvez les séparer à l'aide d'eau chaude, de savon et d'un peu de patience. En aucun cas cela ne doit se produire sur les lèvres ou les paupières! Ne vous grattez en aucun cas le visage et ne vous frottez jamais les yeux lorsque vous manipulez une colle rapide !

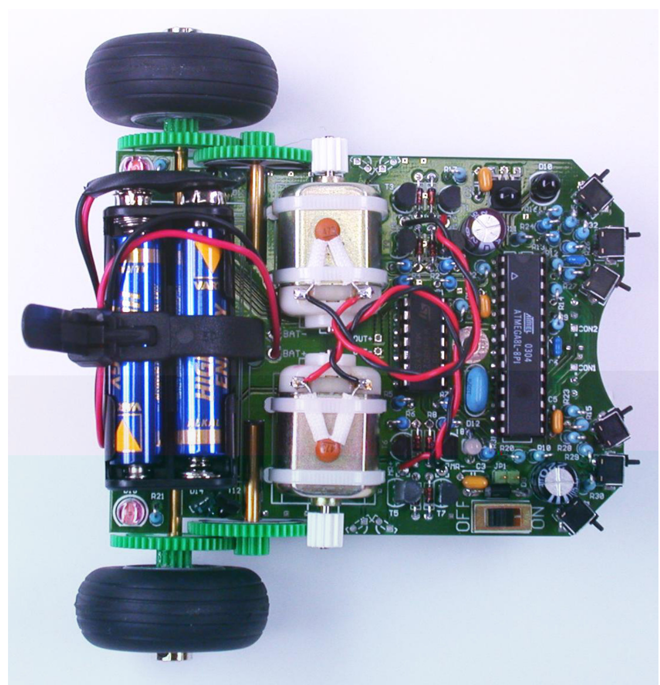
7. Derniers réglages

Lubrifier légèrement les essieux et placez l'engrenage avec le beau motif noir et blanc sur l'essieu court. Placez le pneu sur l'engrenage comportant les 50 et 12 dents. Montez l'ensemble sur l'essieu arrière et fixez-le au moyen d'une bague de façon à ce que l'ensemble puisse tourner facilement.

Déplacez le moteur qui était fixé provisoirement, jusqu'à ce qu'il soit droit, que son pignon morde sur toute la largeur du premier engrenage et que pignon et engrenage tournent facilement (répéter éventuellement l'auto-test complet pour vérifier si tout tourne lors du test moteur).

Lorsque vous avez trouvé la bonne position, tenez fermement moteur et platine et fixez les deux ensemble en mettant quelques gouttes de colle rapide sur le côté dans la fente entre le moteur et la platine. N'oubliez pas que même la colle rapide met parfois quelques minutes à durcir.

Appliquez maintenant de la colle rapide sur l'arête de chaque côté de la demie balle de ping-pong et collez-la sur le dessous de la platine directement derrière les composants de suivi de tracé (voir Fig. 7.1) et laissez sécher.



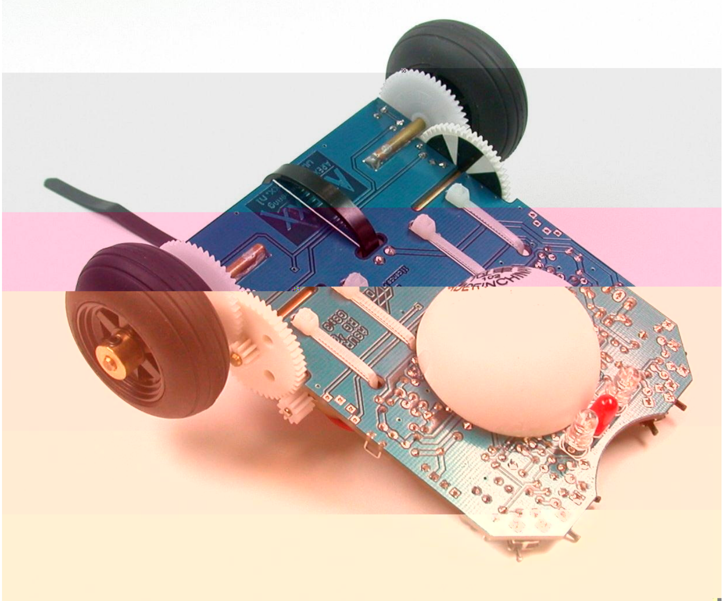


Fig. 7.1.: ASURO fini

Partie III. Informatique

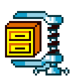
8. Installation du logiciel et premières étapes

Insérez le CD-ROM d’ASURO dans le lecteur. Il démarre automatiquement. Si le démarrage automatique a été désactivé, vous avez la possibilité d’ouvrir le CD avec Windows Explorer. Après la sélection de la langue, vous trouverez tout dans le chapitre « Software » dont vous avez besoin pour l’utilisation d’ASURO. Commencez par installer ces programmes. L’installation du compiler nécessite des droits d’administrateur. Si l’utilisateur actuel ne possède pas ces droits, il doit fermer la session et ouvrir une nouvelle session comme administrateur.

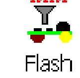
Pendant l’installation du logiciel, les étapes suivantes se dérouleront:

1. Installation de l’outil Flash pour transmettre ses propres programmes sur ASURO
2. Installation d’un éditeur de programme (Programmers Notepad 2, PN2) et d’un compileur (WinAVR)
3. Copie d’un programme de démonstration sur le disque dur
4. Installation des outils MAKE et CLEAN dans le menu de l’éditeur (PN2).

8.1 Windows 

8.1.1 Outil Flash 

Vous pouvez soit copier l’outil Flash dans un répertoire sur le disque dur (p.ex.:C:\Programme\Flash) ou l’exécuter plus tard directement à partir du CD. Dans tous les cas, il est utile de placer un raccourci sur le bureau d’où vous pourrez démarrer l’outil Flash rapidement.



Cliquez sur [Save]

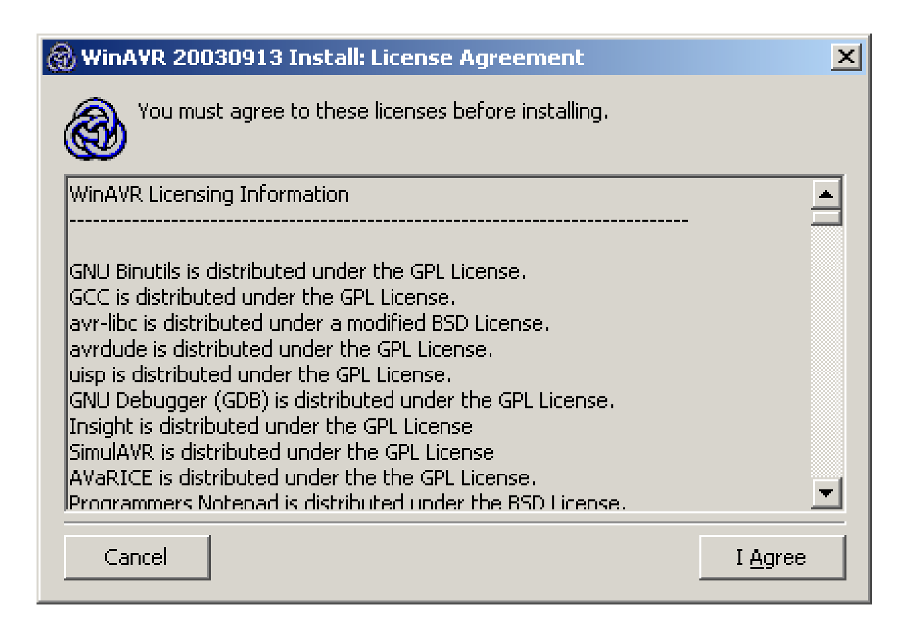
8.1.2 Installation de l’Editeur de programme et du Compileur

L’installation du compileur nécessite des droits d’administrateur (parce que l’enregistrement est modifié pendant l’installation). Si l’utilisateur actuel ne possède pas ces droits, il doit fermer la session et ouvrir une nouvelle session comme administrateur!

Cliquez sur [Install]

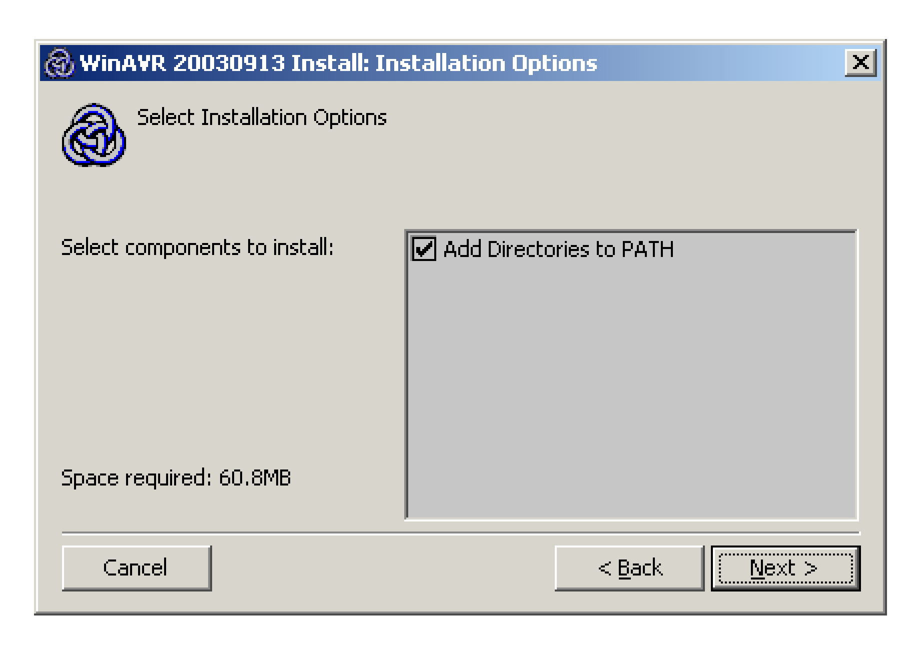


La fenêtre suivante s'affiche:



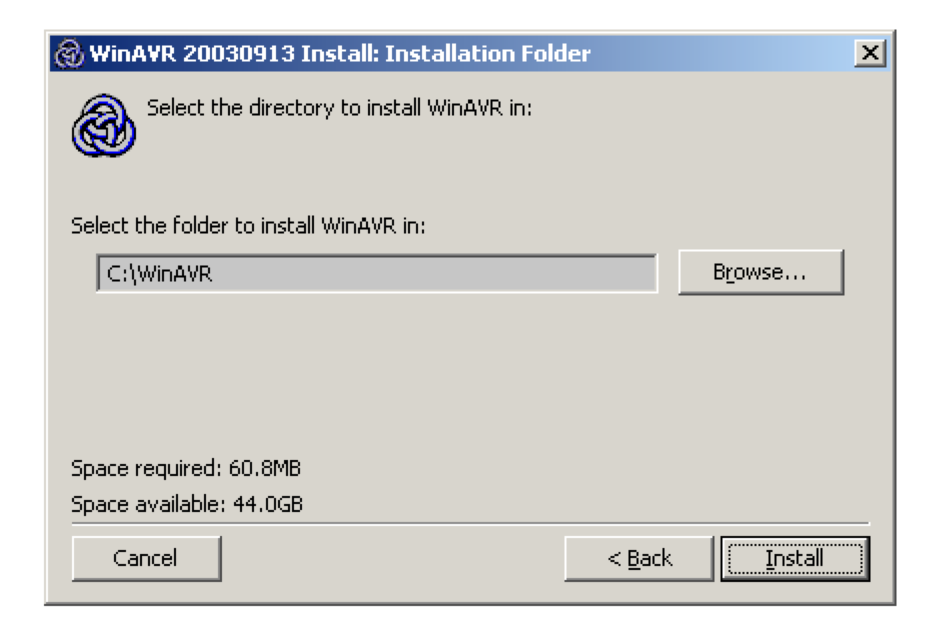
Cliquez sur [I Agree]

La fenêtre suivante s'affiche:



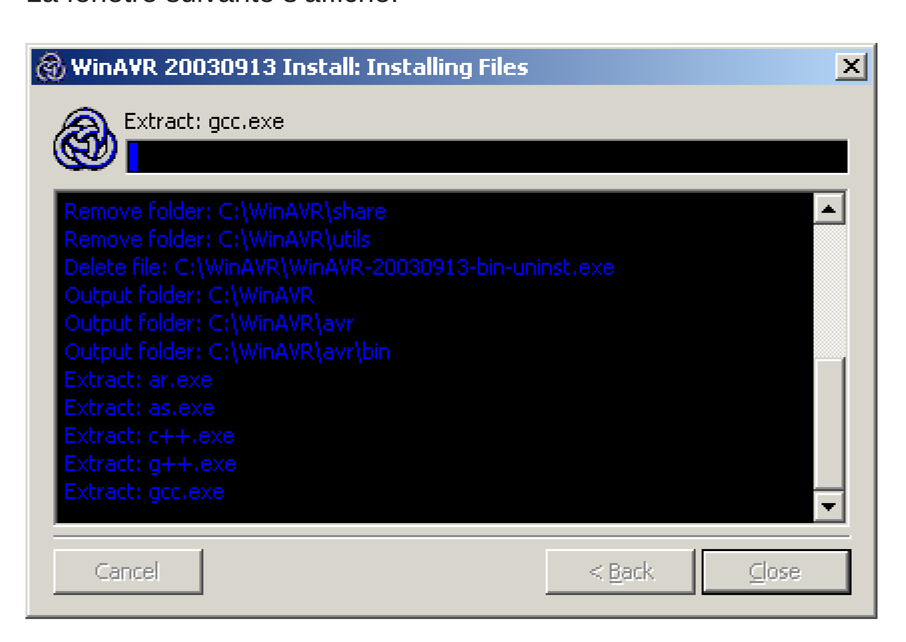
Cliquez sur [Next]

La fenêtre suivante s'affiche:



Cliquez sur [Install]

La fenêtre suivante s'affiche:



Attendez…

…jusqu'à ce que l'éditeur Programmers Notepad 2 (PN2) s'affiche avec le fichier README.txt.

```
Programmers Notepad 2
File  Edit  View  Tools  Window  Help
README.txt
README.txt
WinAVR 20030913
--------------------------
WinAVR (pronounced "whenever") is a suite of executable, open source software
development tools for the Atmel AVR series of RISC microprocessors hosted on
the Windows platform. It includes the GNU GCC compiler for C and C++.

I'm always interested in how you are using the software. If you have a chance,
send an email to me and let me know how you use WinAVR. Thanks!

- Eric B. Weddington <eric@ecentral.com>


Contents:
--------------------------
1.0 What's New
2.0 WinAVR Installation
    2.1 Manifest
    2.2 PATH environment variable
    2.3 Layout
3.0 Toolset Background
    3.1 The Toolset and Open Source
    3.2 Compiler
    3.3 Assembler, Linker, Librarian and More
    3.4 C Library
    3.5 Making Your Software
    3.6 Programming
    3.7 Debugging
4.0 Setting Up a Project
    4.1 Where's the GUI / IDE?
    4.2 Programmers Notepad
    4.3 Make, Makefiles, and the Sample Makefile
    4.4 Building from AVR Studio 3.x
    4.5 Debugging, Simulating, and Emulating
    4.6 AVR Studio Debugging Requirements (COFF Files)
    4.7 Operating Systems
5.0 Finding Help
    5.1 Documentation
    5.2 Online Sources
6.0 Toolset Bugs
    6.1 AVR COFF Bugs
7.0 WinAVR Project
    7.1 WinAVR FAQ
[1:1] : 1178                Ready
```

Fermez maintenant la fenêtre 'Programmers Notepad 2'.

Sur le bureau apparaît le raccourci 'Programmers Notepad 2':



L'éditeur de programme et le compileur sont maintenant installés.

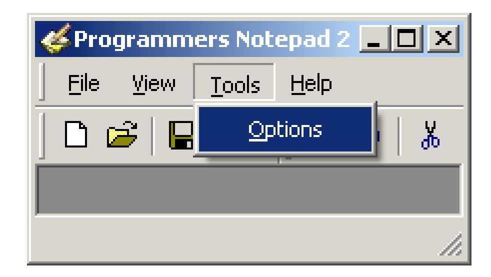
8.1.3. Copie des programmes de démonstration du CD ROM sur le disque dur

Copier le dossier ‚ASURO_src' à partir du CD ROM dans un dossier (p.ex. 'C:\ASURO_src') sur le disque dur.

Mettez les fichiers copiés en surbrillance dans le répertoire cible et faites un clic droit sur la souris pour désactiver la protection d'écriture dans les Propriétés.

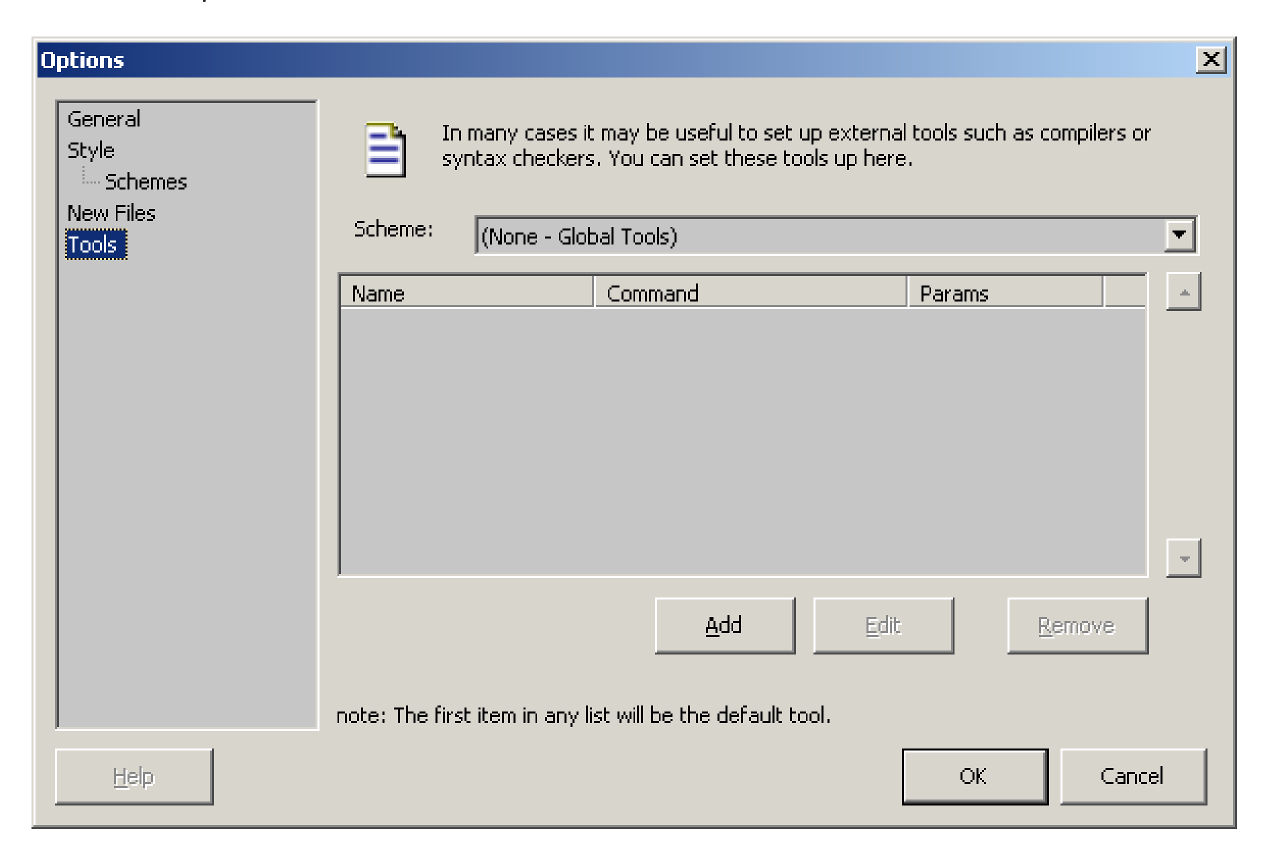
Ajout d'une fonction de compilation dans le menu de l'éditeur de programme

Ouvrez 'Programmers Notepad 2' en double-cliquant sur son raccourci sur le bureau:



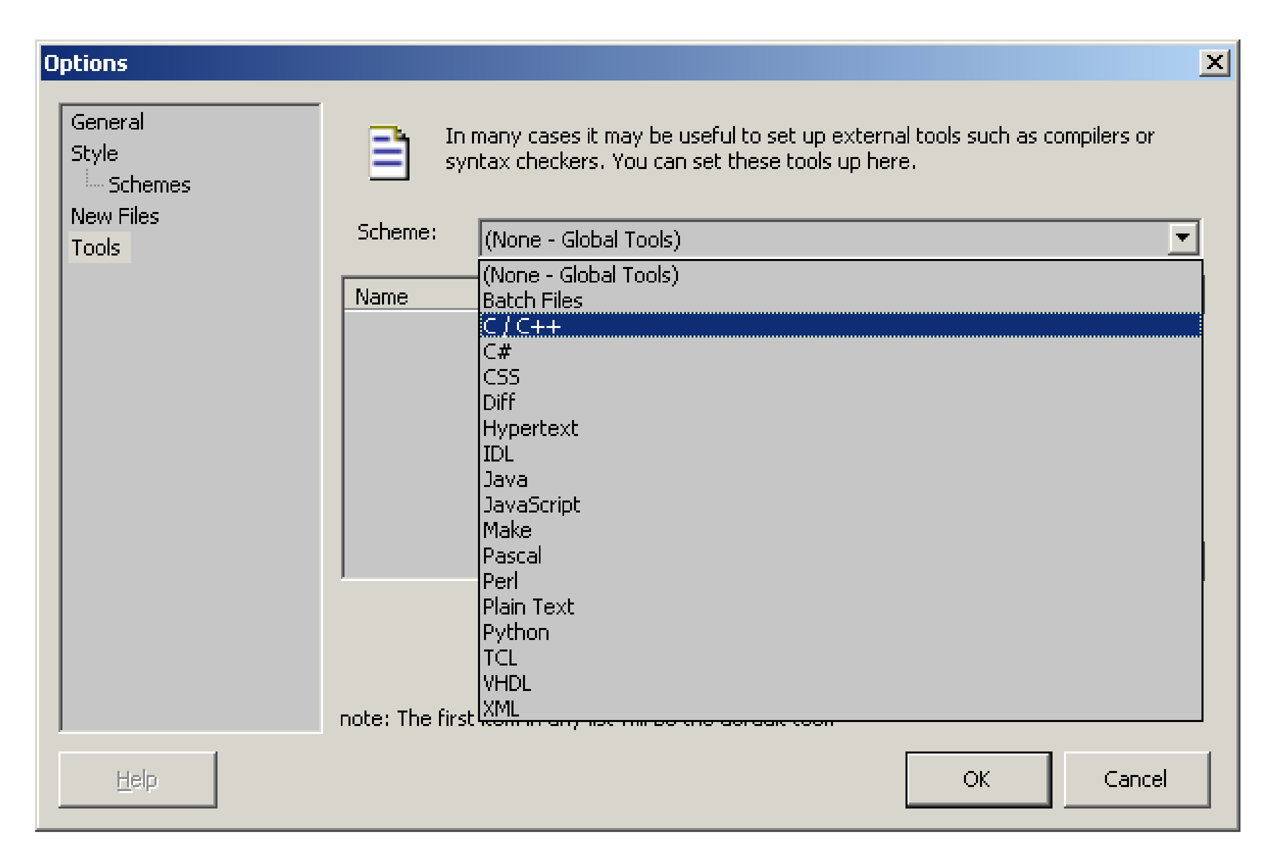
Sélectionnez 'Options' dans le menu 'Tools'.

La fenêtre ‚Options' s'affiche.

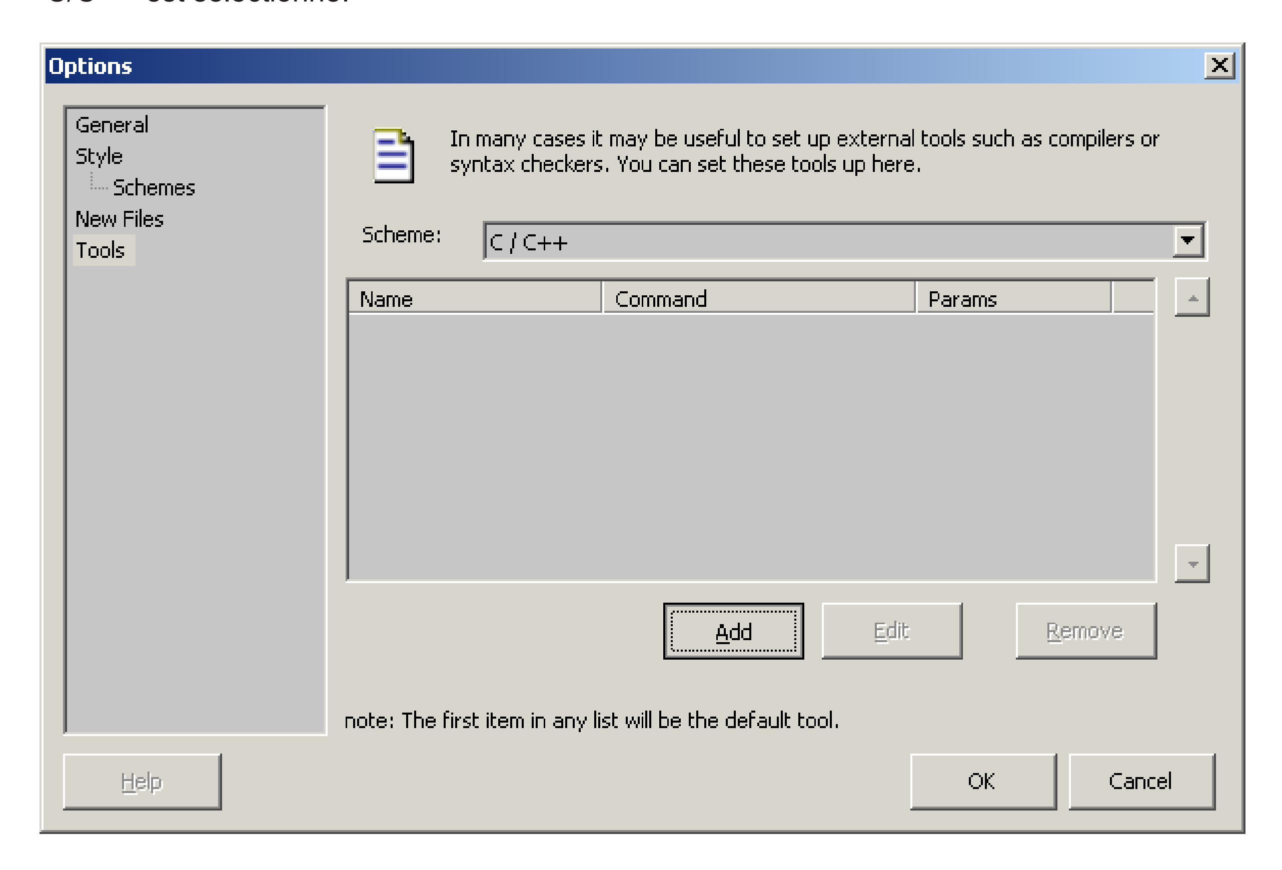


Sélectionnez maintenant Tools.

Sélectionnez ‘C/C++’ dans ‘Scheme’ sur le côté droit.

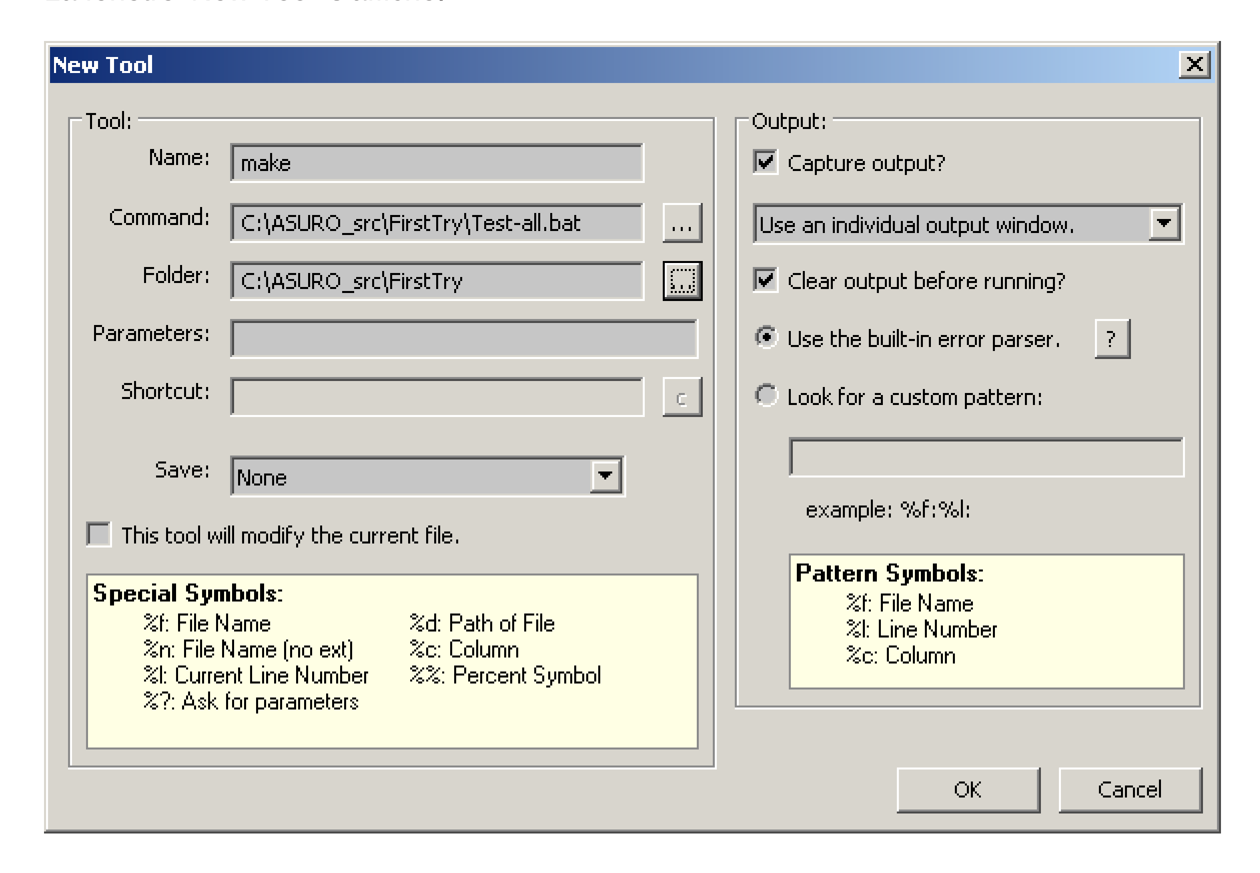


‘C/C++’ est sélectionné.



Cliquez sur [Add] (…pour ajouter un nouvel outil)

La fenêtre ‘New Tool’ s’affiche.



Saisir les réglages suivants ou les sélectionner à l’aide de la touche Parcourir ... :
Name: make
Command: C:\ASURO_src\FirstTry\Test-all.bat
Folder: C:\ASURO_src\FirstTry

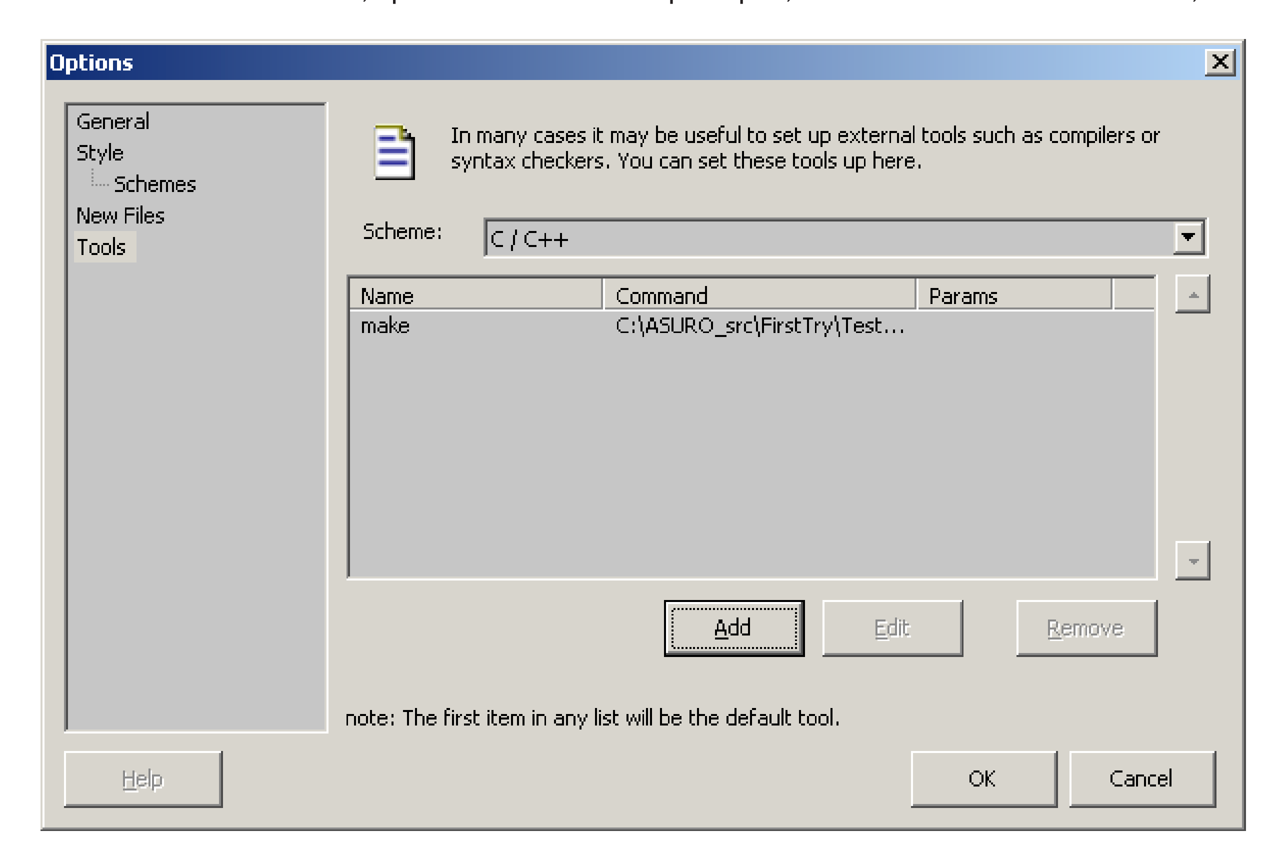
Cliquez sur [OK]

Un nouvel outil PN du nom de ‚Make’ est immédiatement disponible dans le menu principal ‚Tools’.

(Si l’outil est activé, un fichier Batch du nom de Test-clean.bat est exécuté qui compile le programme text.c – en même temps que asuro.c – et génère un fichier text.hex)

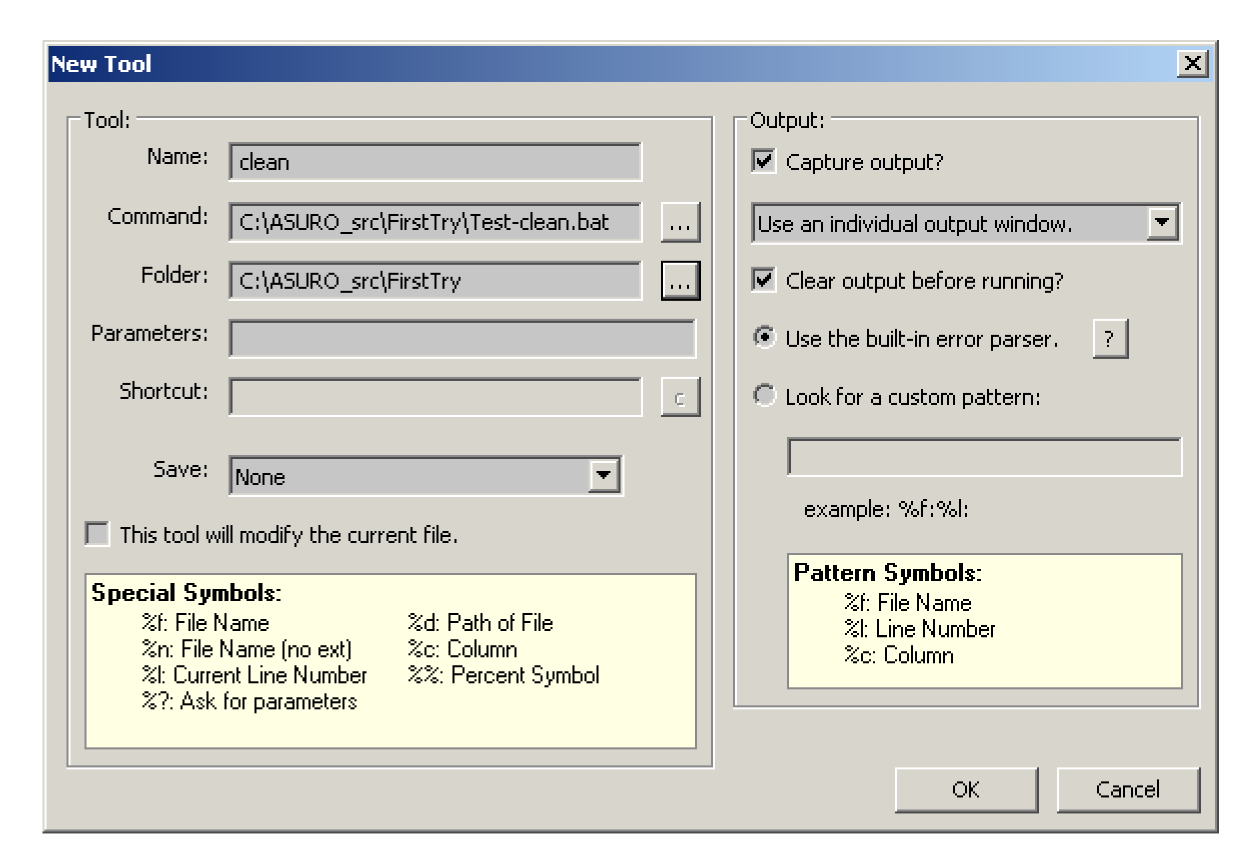
Installation d'une fonction de Nettoyage dans le menu de l'éditeur de programme

Sélectionnez à nouveau ‚Options' dans le menu principal ‚Tools' et encore une fois dans ‚Scheme': „C/C++"



Cliquez sur [Add] pour ajouter un autre outil

La fenêtre ‘New Tool’ s’affiche.

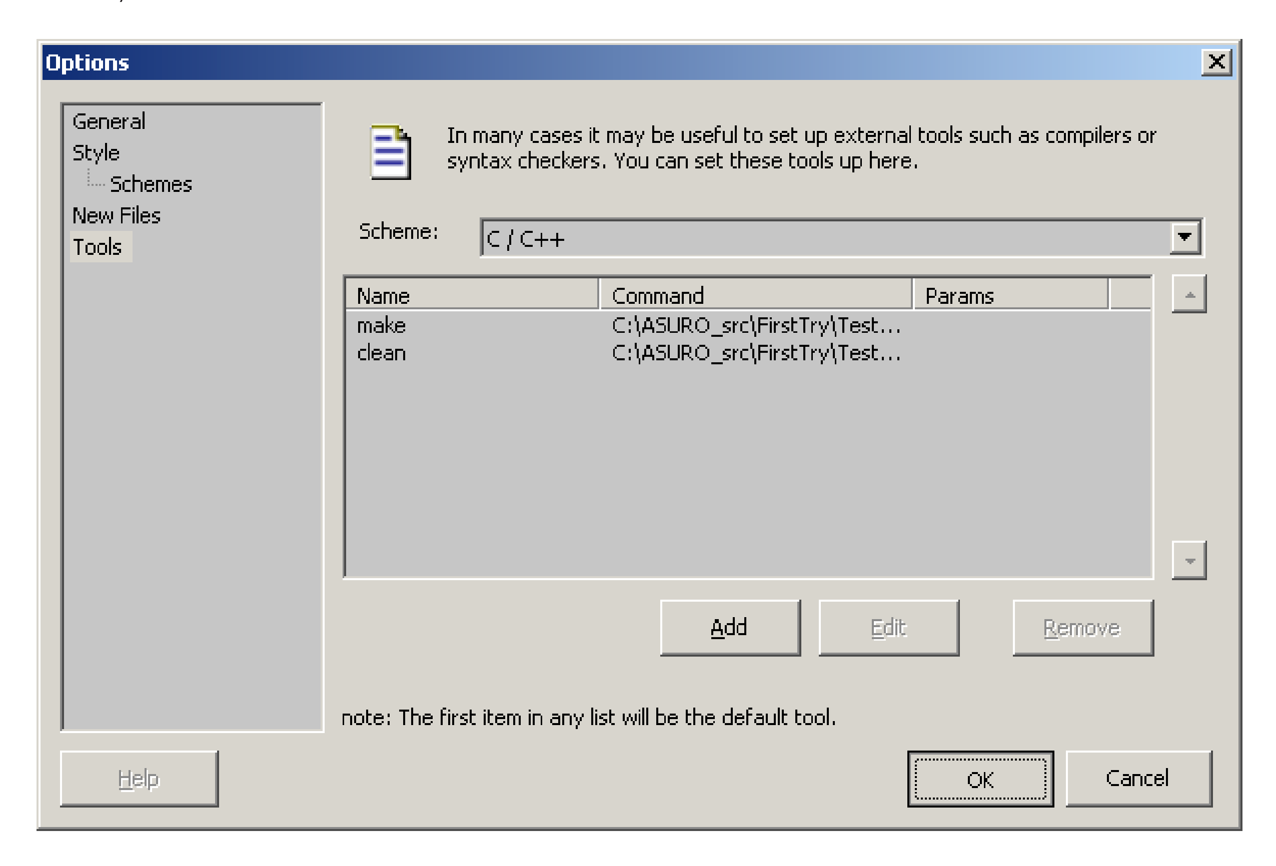


Saisir les réglages suivants ou les sélectionner à l’aide de la touche Parcourir ... :
Name: clean
Command: C:\ASURO_src\FirstTry\Test-clean.bat
Folder: C:\ASURO_src\FirstTry

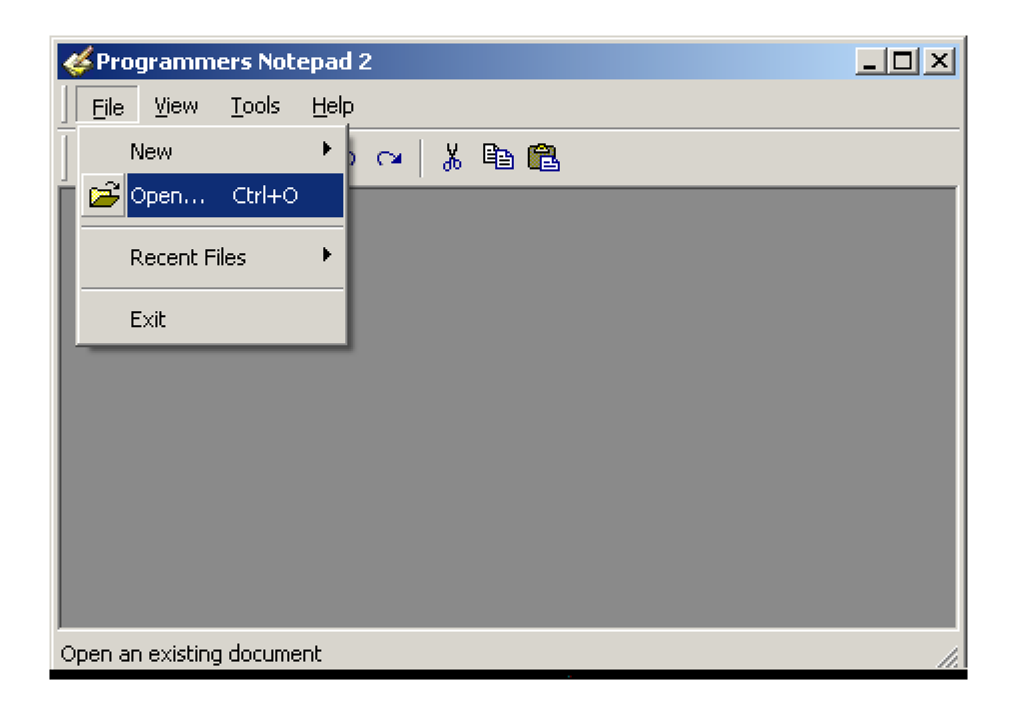
Cliquez sur [OK]

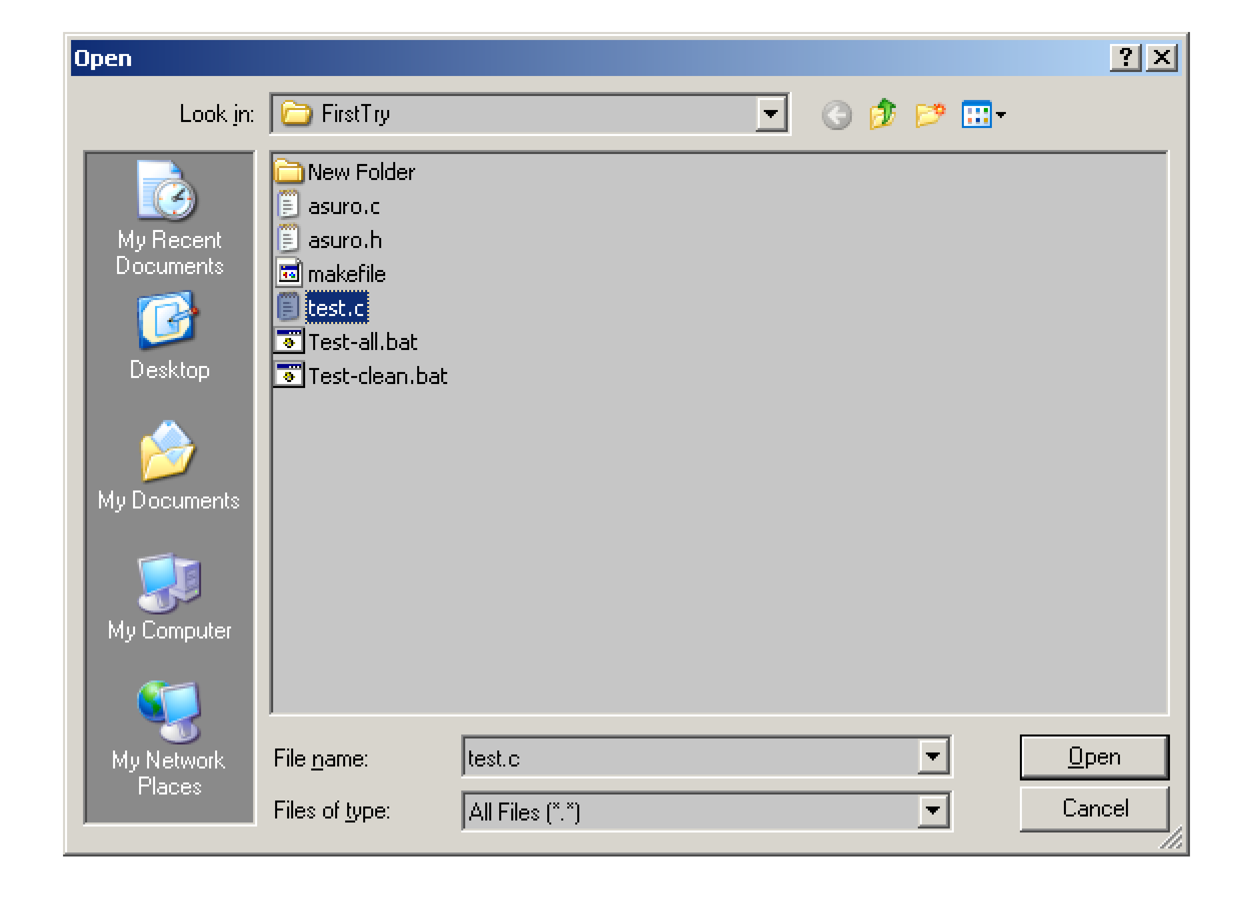
Un nouvel outil PN du nom de ‚Clean’ est immédiatement disponible dans le menu principal ‚Tools’. (Si l’outil est activé, un fichier Batch du nom de Test-clean.bat est exécuté qui efface des fichiers temporaires dans le dossier C:\ASURO_src\FirstTry.

Dans la fenêtre des options, les deux fonctions supplémentaires ‘make’ et ‘clean’ doivent apparaître dans le menu ‚Tools’.



Cliquez sur [OK].

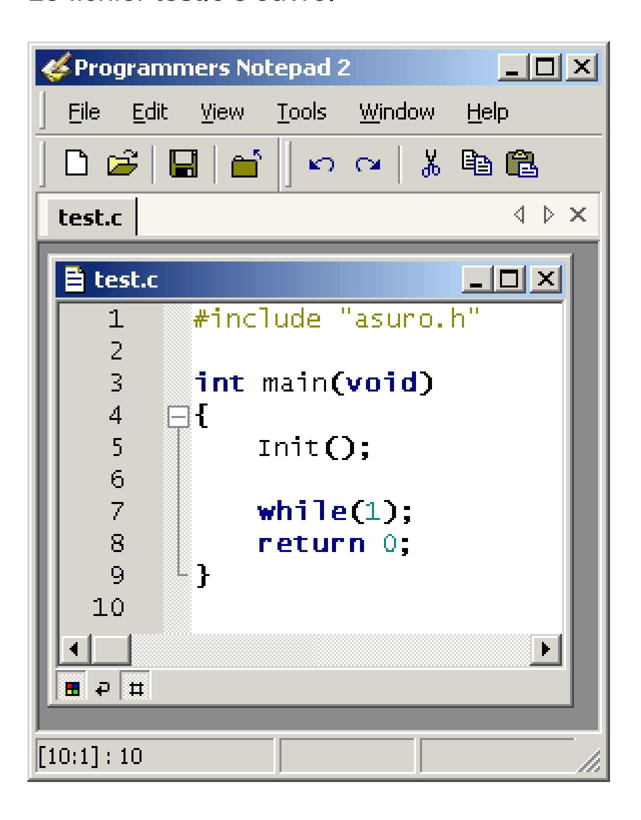




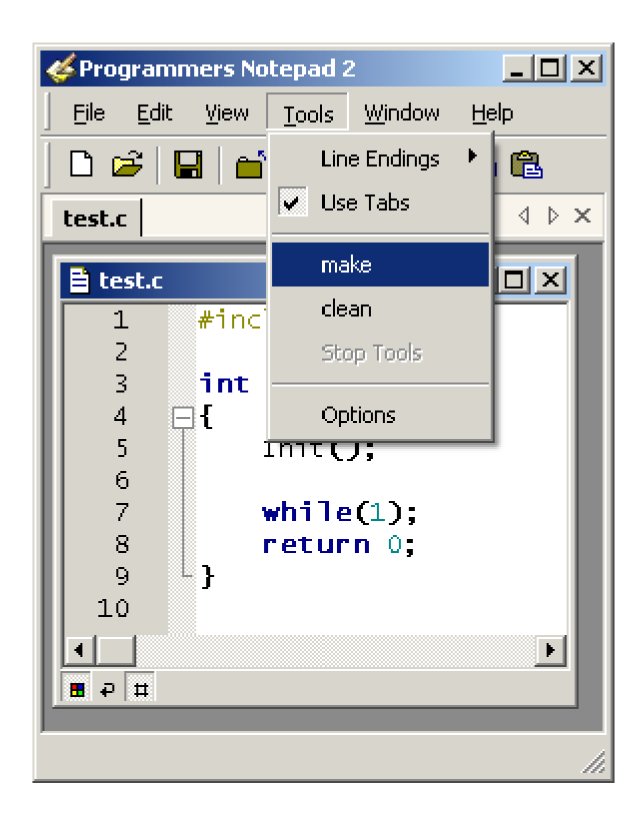
Cliquez sur [Open].

A titre d’essai, ouvrez le fichier ’C:\ASURO_src\FirstTry\test.c’:

Le fichier test.c s’ouvre.



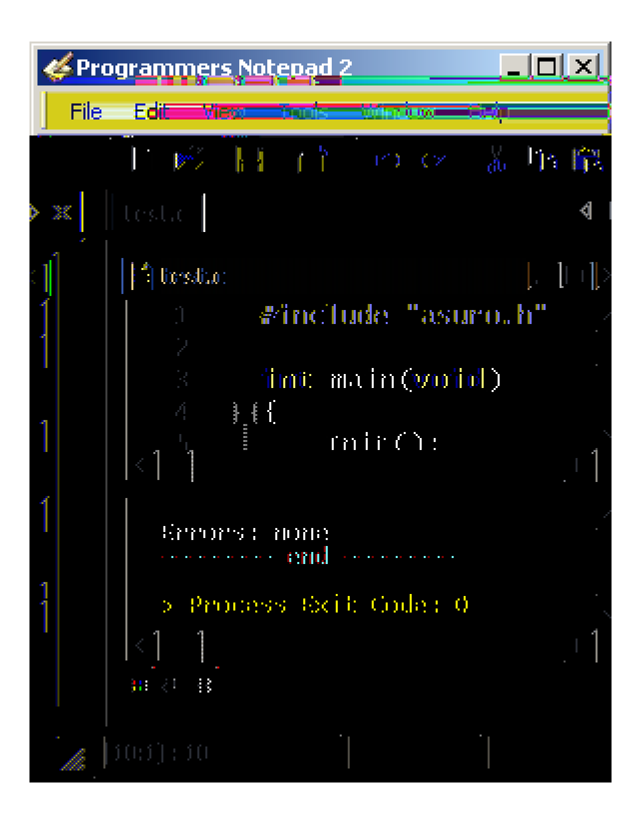
Lorsque vous sélectionnez ‚Tools’…



…vous voyez les deux nouveaux outils ‚make’ et ‘clean’.

Cliquez sur ‚make’

Le fichier test.c (avec asuro.c) sera maintenant compilé…



…et si le programme ne contient pas d’erreurs (ce qui est probable puisqu’il s’agit du programme de démonstration) le message s’affiche: Errors: none.

Que s’est-il passé ?
Un fichier test.hex a été généré à partir du fichier test.c (et asuro.c). Ce fichier représente le programme traduit en langage machine qui est maintenant prêt à être chargé dans la mémoire d’ASURO. Le programme lui-même ne fait rien mais nous en aurons besoin ultérieurement pour tester l’outil Flash.

Comment est-ce que cela a fonctionné ?
La fonction ‘make’ appelle le fichier Batch Test-all.bat (un fichier Batch contient une liste de lignes de commandes qui sont exécutées l’une après l’autre).
Le fichier Test-all.bat exécute la commande ‘make all’. ‘make’ exécute toujours un fichier make qui doit se trouver dans le même dossier (lors de la programmation d’ASURO) que le fichier Test-all.bat .

Un fichier Make est un fichier de texte qui détermine la façon de compiler un ou plusieurs programmes. Cela reste encore relativement simple si le programme n’est traduit qu’à partir d’un seul fichier mais depuis que des systèmes d’exploitation entiers sont écrits en C et que le code est reparti sur plusieurs fichiers qui doivent être traduits et liés dans un ordre précis, même un fichier Make peut devenir très compliqué.
Le terme ‘all’ appelle la particule ‚all’ dans le nom de fichier Make qui signifie qu’il faut traduire un projet complet et non pas seulement certaines parties.

Le fichier Make qui fait partie de notre programme de démonstration est écrit de telle façon qu'il compile un fichier du nom de test.c avec asuro.c (qui contient quelques fonctions pré-définies) et génère un fichier .hex qui peut être chargé tel quel dans ASURO.

Attention! Cela signifie aussi que tant que le fichier Make n'est pas modifié mais seulement copié, votre propre programme doit toujours s'appeler test.c.

Si vous voulez tout savoir sur les fichiers Make (ce qui n'est pas nécessaire pour les premiers pas), vous pouvez consulter la documentation p.ex. sur le site Internet http://www.gnu.org/directory/make.html.

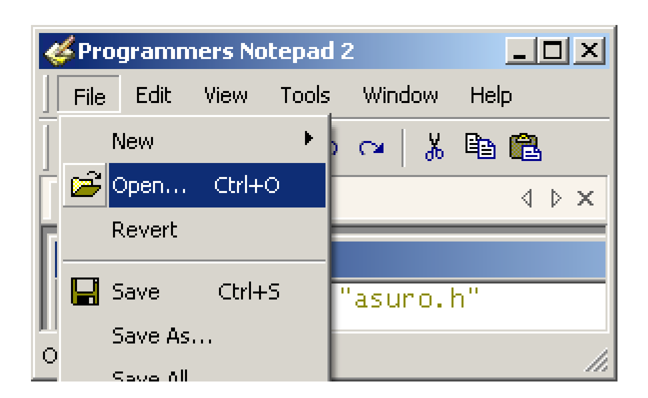
Les bases de la programmation d'ASURO sont expliquées au chapitre 9.

La compilation d'un programme génère quelques « fichiers secondaires » qui ne sont requises que pendant la traduction et deviennent ensuite superflus. L'outil nouvellement créé 'Clean' permet de les effacer.

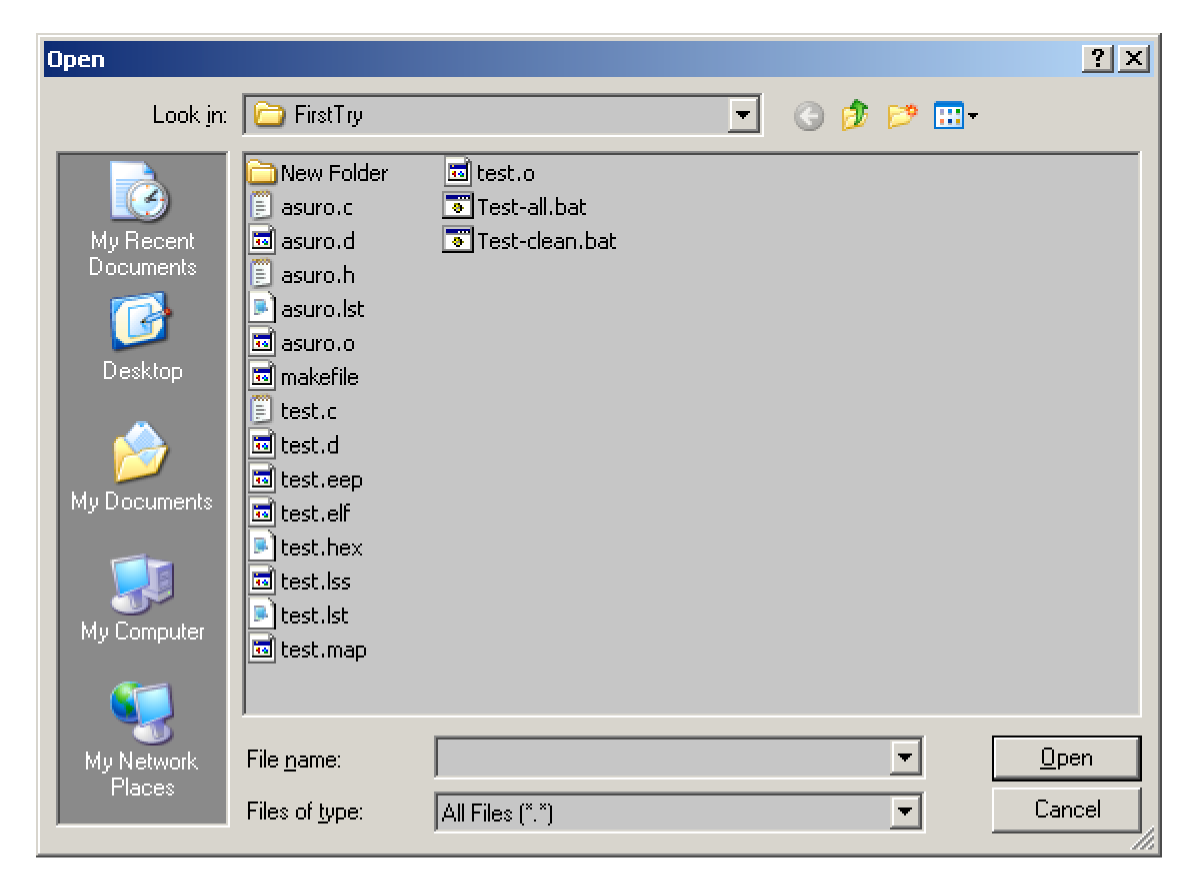
WWW.AREXX.COM



Lors de l'ouverture…

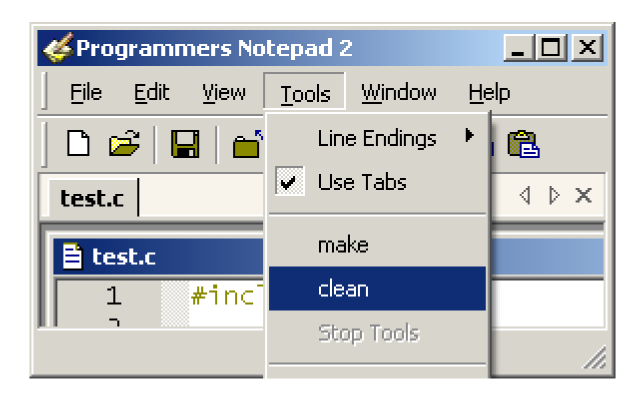


…vous pouvez voir les fichiers générés…

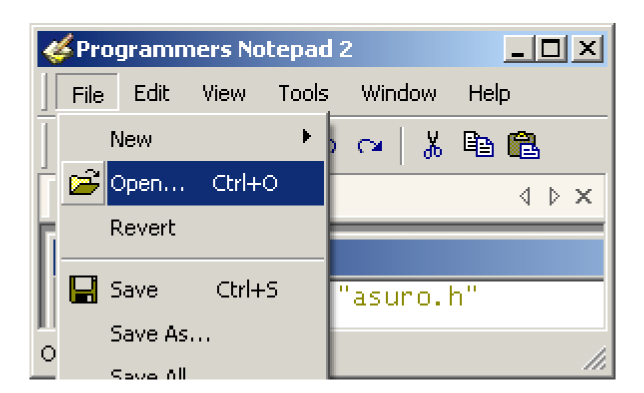


(Cliquez sur [Cancel])

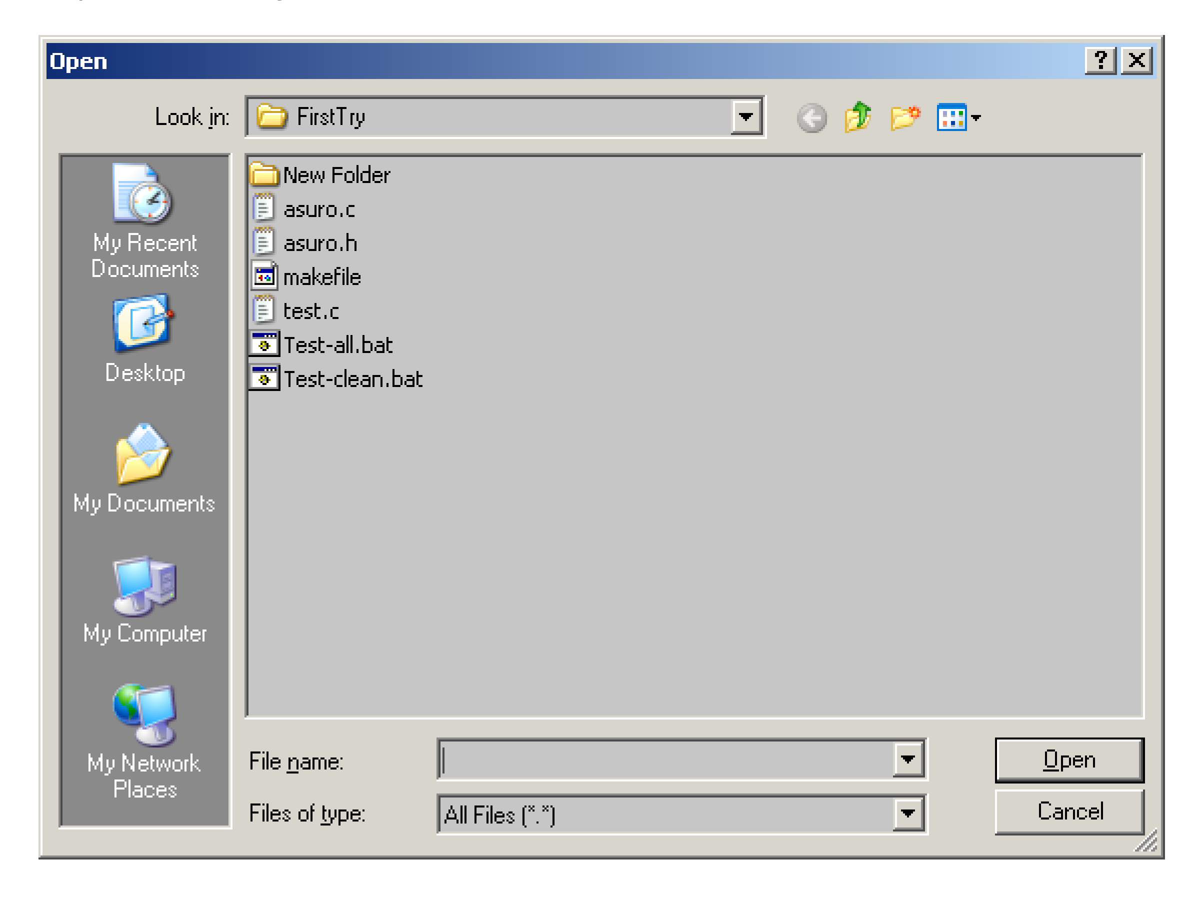
…et après l'exécution de 'clean'…



…vous allez voir…



…que les fichiers générés ont été effacés.



Comment cela s'est-il produit?

L'outil ‚clean' a appelé le fichier Batch Test-clean.bat qui a démarré Make avec le paramètre ‚Clean'. Ceci entraîne l'exécution de la fonction Clean dans le fichier Make et efface tous les fichiers superflus.

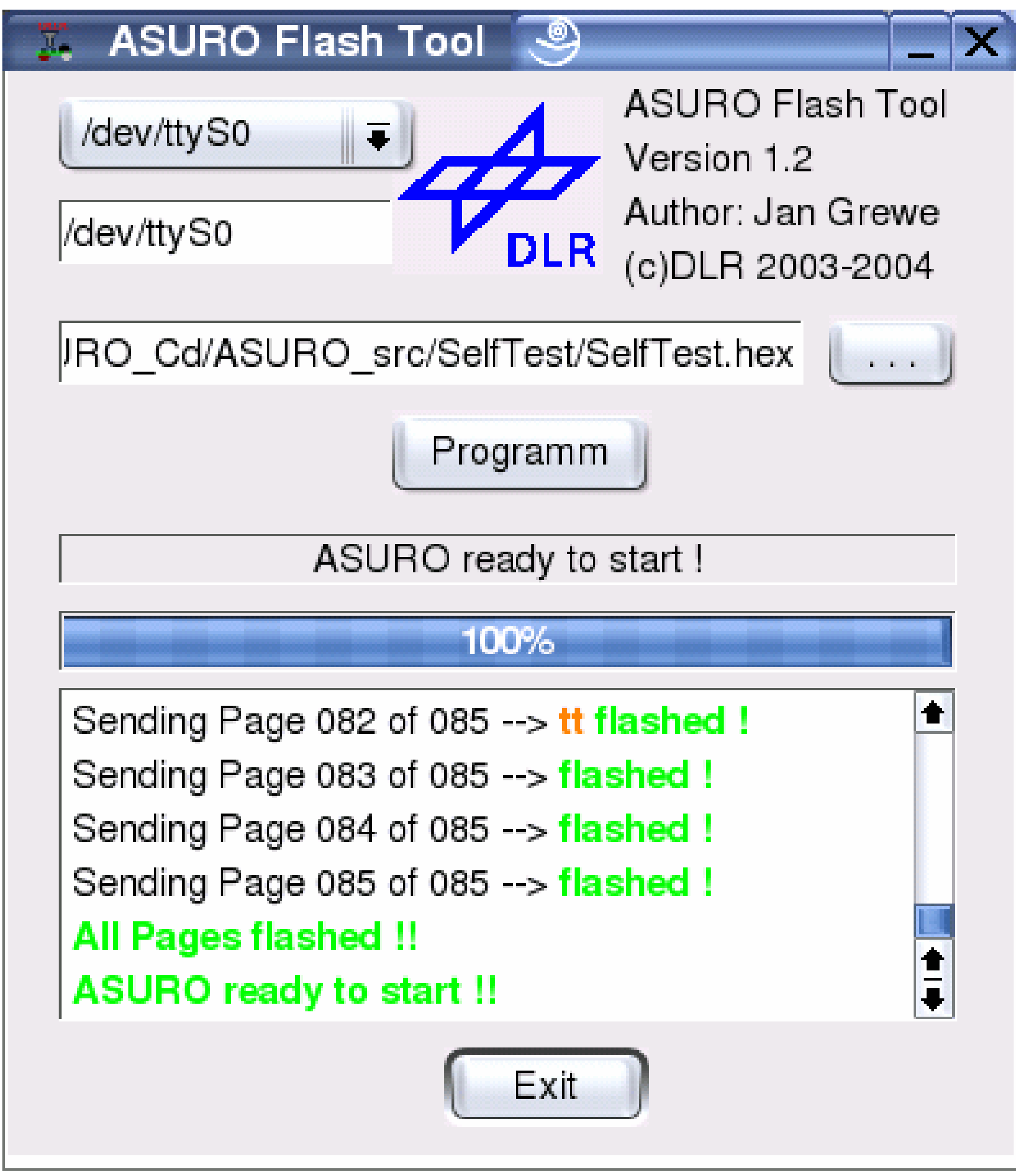
8.2. LINUX

L’installation du logiciel requiert des droits ‚Root’. Soit vous vous déloggez et vous vous loggez à nouveau en tant que Root ou vous ouvrez une shell et vous acquérez des droits Root avec „su“.

8.2.1 Outil Flash

Insérer le CD-ROM ASURO, le monter le cas échéant et copier les deux outils Flash “asurofl ash” et “asurocon” du répertoire “/Linux/Tools/” dans le répertoire “/usr/local/bin”.

Autorisez ensuite l’exécution “chmod a+x /usr/local/bin asurocon asurofl ash”.
Si un “asuroflash” saisi dans une shell n’est pas localisé, il faut ajouter le chemin “/usr/local/bin” dans la variable %PATH-Variable ou appeler le programme avec le chemin complet.



Le premier programme a été traduit sans erreurs.

Fig. 8.1.: Outil Flash

8.2.2 Compiler

Pour installer le compiler Gnu pour processeurs AVR, insérez le CD-ROM ASURO et installez à partir du répertoire “/Linux/Compiler/” au moins les paquets suivants dans l’ordre indiqué:

1. avr-binutils-... .rpm
2. avr-gcc-... .rpm
3. avr-libc-... .rpm

L’installation est plus que simple!
Entrer tout simplement la commande : rpm -i <paket>.rpm dans la machine possédant les droits Root.

Fini!

Les éditeurs Exmacs, Kate ou Kedit conviennent parfaitement. A titre d’essai, vous pouvez copier (comme utilisateur normal) les fichiers de démonstration du répertoire “/ASURO_src/FirstTry/” du CD dans le répertoire Home p.ex. sous “~/ASURO/”.

Ensuite vous ouvrez une shell, vous allez dans le répertoire ci-dessus et vous entrez ‚make’. Si tout a été correctement installé, cela se présentera à peu près de la manière suivante: (voir fig. 8.2)

Befehlsfenster - Konsole

Sitzung Bearbeiten Ansicht Einstellungen Hilfe

```
grewe@linux:~/FirstTry> make all
set -e; avr-gcc -MM -mmcu=atmega8 -I. -g -O3 -funsigned-char -funsigned-bitfield
s -fpack-struct -fshort-enums -Wall -Wstrict-prototypes -Wa,-ahlms=asuro.lst asu
ro.c \
| sed 's,\(.*\)\.o[ :]*,\1.o \1.d : ,g' > asuro.d; \
[ -s asuro.d ] || rm -f asuro.d
set -e; avr-gcc -MM -mmcu=atmega8 -I. -g -O3 -funsigned-char -funsigned-bitfield
s -fpack-struct -fshort-enums -Wall -Wstrict-prototypes -Wa,-ahlms=test.lst test
.c \
| sed 's,\(.*\)\.o[ :]*,\1.o \1.d : ,g' > test.d; \
[ -s test.d ] || rm -f test.d
-------- begin --------
avr-gcc --version
avr-gcc (GCC) 3.3 20030512 (prerelease)
Copyright (C) 2003 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

avr-gcc -c -mmcu=atmega8 -I. -g -O3 -funsigned-char -funsigned-bitfields -fpack-
struct -fshort-enums -Wall -Wstrict-prototypes -Wa,-ahlms=test.lst test.c -o tes
t.o
avr-gcc -c -mmcu=atmega8 -I. -g -O3 -funsigned-char -funsigned-bitfields -fpack-
struct -fshort-enums -Wall -Wstrict-prototypes -Wa,-ahlms=asuro.lst asuro.c -o a
suro.o
avr-gcc -mmcu=atmega8 -I. -g -O3 -funsigned-char -funsigned-bitfields -fpack-str
uct -fshort-enums -Wall -Wstrict-prototypes -Wa,-ahlms=test.o test.o asuro.o   -
-output test.elf -Wl,-Map=test.map,--cref -lm
avr-objcopy -O ihex -R .eeprom test.elf test.hex
avr-objcopy -j .eeprom --set-section-flags=.eeprom="alloc,load" \
--change-section-lma .eeprom=0 -O ihex test.elf test.eep
avr-objdump -h -S test.elf > test.lss
Size after:
test.elf  :
section      size       addr
.text        1422          0
.data           0    8388704
.bss            0    8388704
.noinit         0    8388704
.eeprom         0    8454144
.stab        8364          0
.stabstr     2311          0
Total       12097


Errors: none
-------- end --------
grewe@linux:~/FirstTry> █
```

Fig. 8.2.: Make all

8.3. Flash – L'Outil de Programmation d'ASURO



Pour cela, nous avons besoin du programme Flash (voir fig. 8.3).

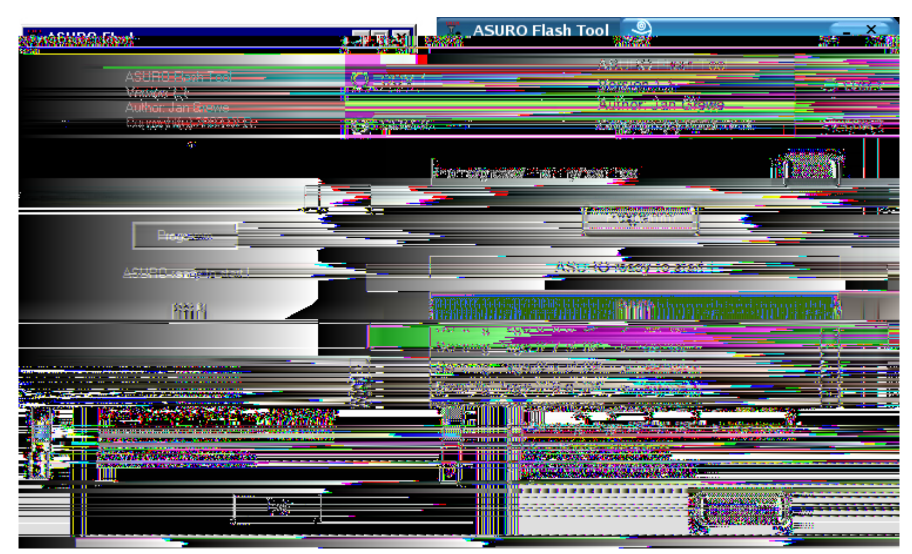


Fig. 8.3.: Outils Flash pour Windows et LINUX

Afin que tout se passe bien, vous devez évidemment connecter l'interface RS232 ou le transmetteur USB-IR. Démarrez ensuite le programme et sélectionnez l'interface qui a fonctionné lors de la mise en service. Sélectionnez le fichier « Test.hex » du répertoire "C:\ASURO_src\FirstTry" (ou ~/ASURO/).

Préparez ASURO (fini et prêt à l'emploi) et cliquez sur le bouton Programme dans l'outil Flash. ASURO doit être en contact visuel avec le transmetteur IR (env. 50cm de distance entre le transmetteur IR et ASURO, les deux faces d'implantation sont dirigées l'une vers l'autre et aucun obstacle ne se trouve dans le chemin de la lumière).

Mettez ASURO sous tension (S1 sur ON) avant que la barre d'état ne soit complètement arrivée sur la droite. Si vous n'avez pas été assez rapide ou si la prise de contact ne s'est pas correctement déroulée, éteignez ASURO, cliquez de nouveau sur Programme et mettez ASURO sous tension.

Si la prise de contact a réussi, vous voyez sur l'indication d'état et dans la fenêtre du protocole que le fichier Test.hex est transmis sur ASURO où il est sauvegardé dans la mémoire Flash du processeur afin d'être toujours disponible, même après la coupure de l'alimentation.

Si le processus s'est correctement déroulé, vous devez éteindre et remettre ASURO sous tension afin de démarrer le programme. Le programme récemment écrit est exécuté et la LED verte s'illumine brillamment.

8.3.1. Fonctionnement du programme Flash

Dès que le programme Flash est exécuté, l'ordinateur essaie d'établir une liaison avec ASURO pendant 10 secondes. Si vous mettez ASURO sous tension, la LED d'état s'allume en 2 couleurs pendant env. 1 seconde. Ceci est la phase Boot. ASURO vérifie si le PC a un nouveau logiciel pour lui qu'il charge le cas échéant. Après une nouvelle mise hors/sous tension, ce logiciel démarrera.

8.4. Erreurs Flash

Les erreurs suivantes peuvent se produire pendant le flashage:

- “c” Checksum Error. ASURO a reçu d’autres données que celles envoyées par le PC.
  Ceci peut être provoqué par des éclairages parasites (p.ex. des néons), des petites coupures pendant le contact visuel et autres.

- “t” Timeout. Le contact visuel avec ASURO a été interrompu.

- “v” Verify Error. ASURO a écrit des données erronées dans sa mémoire Flash.
  Cela ne devrait normalement pas se produire et cela indique que la mémoire non-volatile (Flash-EEPROM) est en fin de vie ce qui n’arrive typiquement qu’après 10.000 programmations. Après 10 tentatives vaines de corriger l’erreur, le flashage est arrêté.



Si des erreurs Checksum se produisent fréquemment pendant le flashage, il est souvent suffisant d’éteindre ou de baisser un peu la lumière de la pièce, notamment quand il s’agit de tubes fluorescents.



Appuyez toujours d’abord sur le bouton Programme et mettez ASURO sous tension ensuite sinon un téléchargement de logiciel n’est pas possible !

8.5. Votre premier programme personnel

Avant de passer à un petit tour d’horizon de la programmation C, voici un premier petit programme personnalisé. A cet effet, nous chargeons le fichier test.c du répertoire C:\Mes fichiers\ASURO_src\FirstTry au moyen du bloc-note Programmers Notepad (Linux : ou un autre éditeur):

```
#include “asuro.h”
int main(void) {
Init();
while(1);
return 0;
}
```

Pour les premiers essais, il est impératif que le programme porte toujours le nom de test.c car le fichier Make de démo fourni (un fichier qui décrit la traduction d’un programme) est construit de cette façon. Aussi, il est plus simple de continuer l’écriture à partir d’un exemple existant. Plus tard, vous pourrez concevoir vos propres programmes et écrire vos propres fichiers Make.

Le programme chargé sera modifié comme suit (Attention à l'orthographe exacte ! Veillez aux majuscules et minuscules !):

```
#include "asuro.h"
int main(void) {
Init();
StatusLED(RED);
while(1);
return 0;
}
```

Sélectionner ensuite de nouveau Make dans le menu Tools (sous Linux: Taper „make" dans une shell dans le répertoire "~/ASURO/" ou bien configurer l'éditeur en conséquence et attendre la compilation jusqu'à ce qu'il n'y ait plus de nouveaux messages dans la fenêtre d'état).
Vérifier si Process Exit Code: 0 s'affiche en bas de la fenêtre d'état. Cela signifie que le compileur a compris et traduit le programme.
Si un autre code s'affiche, vous devez chercher l'erreur à l'aide des messages d'erreur.

Dans la plupart des cas, il suffit de commencer la recherche dans la ligne où, d'après la fenêtre d'état, la première erreur est apparue. Le numéro de ligne à laquelle se trouve le curseur s'affiche tout en bas à gauche dans l'éditeur.

Si la compilation s'est déroulée sans erreurs, le nouveau programme est prêt au flashage. A cet effet, il faut connecter le transmetteur IR, démarrer l'outil Flash, sélectionner le fichier test.hex et la bonne interface COM, mettre ASURO en contact visuel avec le transmetteur IR, cliquer sur le programme, mettre ASURO sous tension et attendre la transmission du programme.

Si, d'après la fenêtre d'état, la transmission s'est déroulée sans heurts, éteindre ASURO, une seconde de suspense et …. BRAVO… la LED d'état s'allume en rouge.
Afin d'éviter d'écrire des lignes de programmation sans avoir les connaissances nécessaires, nous vous recommandons de lire le chapitre suivant avant de vous lancer dans d'autres expériences.

9. C pour ASURO

Ce chapitre traite du langage de programmation C. Seuls les parties de C nécessaires à la programmation d'ASURO seront expliquées. Il ne s'agit en aucun cas d'une introduction complète dans le langage C. Pour cela, il existe des ouvrages spécialisés. C a été choisi comme langage de programmation parce qu'il est largement répandu et qu'il existe au moins un compileur C pour presque chaque processeur.

Pour ASURO, nous avons retenu le compileur GNU-C car il s'agit d'un programme gratuit (freeware) qui génère quand-même un code bien optimisé pour l'ATmega8, le processeur d'ASURO.

Si vous savez déjà programmer en C, allez directement au chapitre 9.2 car ce qui suit ne présente absolument aucun intérêt pour vous. Nous ne donnons ici que les éléments de langage les plus indispensables pour transmettre de la façon la plus simple les connaissances qu'il faut absolument avoir pour le fonctionnement d'ASURO.

Et ne vous en faites pas : si vous pensez consciencieusement à vos parenthèses et points-virgules, le langage C n'est pas si difficile que cela.

9.1. Bases de la Programmation en C

9.1.1. Généralités

Le processeur traite toujours un programme C expression après expression du haut vers le bas[2]. Une exécution simultanée de deux commandes n'existe pas, en tout cas pas dans le processeur d'ASURO.

Il faut donc réfléchir d'après ce principe : Une commande après l'autre. Les espaces au début des lignes dans les exemples ne sont pas forcément nécessaires mais très utiles lorsqu'il faut maintenir une certaine clarté dans des programmes plus longs.

Chaque expression en C se termine par un « ; » ce qui permet au compileur de différencier les expressions.

Si vous voulez réunir plusieurs expressions ce qui sera nécessaire pour les fonctions, boucles ou conditions (nous y reviendrons plus tard), le bloc d'expressions est encadrés d'accolades (« {« , « } »).

Exemple:

```
#include "asuro.h"
int main (void) {
/* Tout ce qui sera marqué ici doit rentrer dans un bloc */
}
```

---

1 P.ex.: Brian W. Kerninghan, Dennis M. Ritchie: "Programmieren in C", Hanser Verlag, ISBN 3-446-15497-3

2 Des méthodes qui interviennent dans le déroulement séquentiel des commandes, sont appelés „Flux" et seront expliqués plus loin dans le chapitre

Si vous voulez exclure certaines lignes du code, vous devez commencer le bloc de commentaire par “/*” et le terminer par “*/ ”. Pour n’exclure qu’une seule ligne, il suffit de placer un “//“ devant la ligne[3] en question. De cette façon, le compileur ignore les parties exclues. Cela permet d’insérer des commentaires dans le programme sans perturber la traduction.

9.1.2. Variables et Types de données

Les variables sont des „récipients“ pour les données. Au cours d’un programme, vous pouvez les décrire, les lire et les modifier. Pour pouvoir utiliser une variable, vous devez d’abord la déclarer. Ceci détermine le type de la variable et éventuellement aussi sa valeur initiale. Le type définit le genre de nombres vous pouvez mémoriser dans la variable (nombres entiers, nombres entiers positifs, fractions décimales…).

Le nom d’une variable doit commencer par une lettre (« _ » est également considéré comme une lettre) et peut contenir des chiffres mais pas de caractères spéciaux. Il faut différencier entre les minuscules et majuscules ; donc x et X sont des variables différentes.
Traditionnellement, les noms de variables sont donnés en minuscules. Les désignations suivantes sont déjà réservées et ne peuvent pas être utilisées comme noms de variables:

auto	***default***	***float***	***long***	***sizeof***	***union***
break	***do***	***for***	***register***	***static***	***unsigned***
case	***double***	***goto***	***return***	***struct***	***void***
char	***else***	***if***	***short***	***switch***	***volatile***
const	***enum***	***int***	***signed***	***typedef***	***while***
continue	***extern***				

Les types de données suivantes présentent un intérêt pour la programmation d’ASURO :

Typ	Plage de valeur	Observation
char	-128 ... +127	valeur un byte; peut prendre un caractère du bloc de caractères
uncaractèred char	0 ... 255	char sans caractère
int	-32768 .. +32767	magnitude minimale signée
uncaractèred int	0 ... 65535	int magnitude minimale non signée
float		valeur de simple précision

3 “//” est un caractère de commentaire selon le standard C++. Comme le compileur utilisé est en fait un compileur C++, cela fonctionne mais peut provoquer des messages d’erreurs chez d’autres compileurs

La déclaration se fait soit „hors“ de la fonction main() comme variable globale (cela signifie que la variable s’applique à la totalité du programme), à l’intérieur de la fonction main() (dans ce cas, elle ne s’applique qu’au code de programmation qui se trouve dans la fonction main()) ou à l’intérieur de sa propre fonction (dans ce cas, elle ne s’applique qu’à celle-ci).

Le plus belle variable ne sert à rien si l’on ne sait pas comment on peut entrer ou sortir des données.

Il suffit d’une affectation pour entrer des données:

```
a=17; // a a maintenant la valeur de 17
```

ou d’un calcul:

```
a=17+23; // a est maintenant 40
b=a+3; // b est maintenant 43
b=b*2; // b est maintenant 86
```

Et maintenant dans tout le programme:

```
#include “asuro.h”
int main(void) {
        int i; // i peut prendre des nombres entre -32768 et 32767
        caractère char; caractère // peut prendre des nombres entre -128 et 127
        //
        i=3;
        caractère =17+i; // caractère est maintenant 20
        i=i/2; // Division par 2, il faut toujours arrondir; i est donc 1!
        return 0;
}
```

Il existe quelques abréviations pratiques. On peut aussi écrire

```
i=i+1;
```

comme:

```
i++;
```

et

```
i=i-1;
```

correspond à:

```
i--;
```

9.1.3. Directives de Compilation

Vous vous êtes certainement déjà demandé à quoi servait le #include “asuro.h”. La directive #include signifie simplement que le texte qui se trouve dans le fichier indiqué, est inclus dans le programme et sera traduit lors de la compilation. Dans le cas présent, certaines routines qui sont nécessaires au fonctionnement du robot, sont rendues disponibles.

Une autre directive importante (et il y en a d’autres qui dépasseraient le cadre de cet ouvrage) est ce que l’on appelle le remplacement de texte. Celui-ci a la forme

```
#define NAME texte de remplacement
```

et est essentiellement utilisé pour la définition de constantes.

Si le symbole NAME apparaît dans le code, il est remplacé automatiquement par le texte de remplacement.
Le NAME dans #define revêt la même forme que les noms de variables. En langage C, il est devenu usuel d’utiliser des majuscules pour NAME dans #define.

Exemple:

```
#include “asuro.h”
#define VALEURINITIALE 33
int main(void) {
	int i;
	i=VALEURINITIALE; // i est maintenant 33
	return 0;
}
```

Au fait, les directives de compilation ne sont pas suivies d’un point-virgule!

9.1.4. Conditions

Il est souvent nécessaire de faire exécuter des commandes lorsque certaines conditions ont été remplies. A cet effet, nous avons besoin de structures de contrôle. La plus simple qui permet de formuler des décisions est l’expression “if-else”.

La syntaxe suivante s’applique:

```
if (condition)
	Bloc d’expression 1
else
	Bloc d’expressions 2
```

Le processeur vérifie si la condition est vraie. Si elle l’est (donc inégal à 0) le bloc d’expression 1 est exécuté, sinon ce sera le bloc optionnel d’expression 2.

Si vous voulez prendre une décision parmi plusieurs options, vous pouvez utiliser les expressions “else if” :

```
if (Condition 1)
        Bloc d’expressions  1
else if (Condition 2)
        Bloc d’expressions  2
else if (Condition 3)
        Bloc d’expressions  3
else if (Condition 4)
        Bloc d’expressions  4
else
        Bloc d’expressions  5
```

Les conditions suivantes sont possibles:

Opérateur	SIGNIFICATION
==	Comparaison logique avec égal à
!=	Comparaison logique avec n’est pas égal à
<	Comparaison logique avec plus petit que
>	Comparaison logique avec plus grand que
<=	Comparaison logique avec plus Petit ou égal à
>=	Comparaison logique avec plus grand ou égal à

Exemple:

```
#include “asuro.h”
int main(void) {
        Init ()
        while (1) {
        if (PollSwitch()>0) {StatusLED (RED);}
        else {StatusLED (GREEN);}
        }
}
```

Si l’un des capteurs de collision est actionné, la diode d’état s’allume en rouge, sinon elle est verte. Les autres éléments seront expliqués plus loin.

Dans le langage C “1” signifie « vrai » et “0” « faux ». La condition

```
if (0) {StatusLED(RED);}
```

signifie que StatusLED (RED) ne sera jamais exécuté.

9.1.5. Boucles

Les boucles permettent d'exécuter plusieurs fois des expressions.
Dans la boucle „while“, une condition est évaluée. Si la condition est vraie, le bloc d'expressions est exécuté et la condition est vérifiée à nouveau jusqu'à ce qu'elle devienne fausse.

```
while( Condition)
  Bloc d'expressions
```

Exemple:

```
#include “asuro.h”
int main(void) {
        Init ()
        MotorDir(FWD,FWD);              // Les deux moteurs avancent
        MotorSpeed(120,120);            // Les deux moteurs à demie puissance
        StatusLED(GREEN);               // Commuter la diode d'état sur vert
        while (PollSwitch()==0) {       // Tant qu'il n'y a pas eu de collision...
                SerWrite(“Alles OK!\n”,10); // ... Explosion de joie
        }
        MotorSpeed(0,0);                // Collision! Arrêt immédiat!
        StatusLED(RED);                 // Commuter la LED d'état sur rouge
        while (1) {
                SerWrite(“Aua!\n”,5);   // Profonde tristesse!
        }
}
```

L'expression “for (expr1, epr2, expr3)” équivaut à :

```
expr1;
while (expr2) {
        Bloc d'expressions
        expr3;
}
```

La boucle „for“ est normalement utilisée comme boucle de comptage

```
for (i = 0; i < n; i++)
. . .
```

Exemple:

```
#include “asuro.h”
int main(void) {
        Init ()
        int zaehler;                                // Déclarer variable pour le comptage
        for (zaehler=0; zaehler<10; zaehler++) {    // répéter dix fois:
                SerWrite(“Los geht's!\n”,12);       // Envoyer “C'est parti!”
        }
        MotorDir(FWD,FWD);              // Les deux moteurs avancent
        MotorSpeed(120,120);            // Les deux moteurs avancent à demie puissance
        while (1) {                     // Ensuite ne plus rien faire!
        }       }
```

“while(1)” équivalent de “for(;;)” est une boucle sans fin car la condition pour l’arrêt ne sera jamais fausse (donc 0).

Une autre boucle est la boucle „do“.

```
do
  Bloc d’expressions
while( Condition);
```

A l’opposé de la boucle „while“, la condition est vérifiée à la fin du bloc d’expressions. Cette boucle se déroule au moins une fois.

9.1.6. Fonctions (Templates)

Les définitions de fonctions revêtent toujours la forme suivante

Type de fonction Nom de la fonction (ParamètreType 1 Nom du paramètre 1,
Type de paramètre 2 Nom du paramètre 2, ...)

Super! Des définitions de fonctions! Et cela sert à quoi??? C’est très pratique mais aussi plus compliqué et on peut le lire aussi un peu plus tard...

Il arrive fréquemment que des parties se répètent en différents endroits du programme. Soit il faut les réécrire à chaque fois (très ennuyeux et complètement confus) ou on déclare une fois une fonction.

Souvent on aimerait affecter aussi une ou plusieurs valeurs à une fonction. Ainsi une fonction (personnelle) de Avanceunpeu() est tout simplement plus sympa si on peut lui dire la vitesse, la durée ou la trajectoire. Ceci se fait au moyen des paramètres.

Il arrive aussi parfois qu’une fonction renvoie une valeur comme c’est le cas d’une fonction Combiendecapteurssontenfoncés (). Ceci se fait à l’aide de la valeur de retour de la fonction qui est générée quelque part à un moment donné dans la fonction et renvoyée par l’expression return. C’est pourquoi chaque fonction se termine par return; ou return CHIFFRE;.

La fonction main () est particulière car elle constitue le point d’insertion dans un programme. Dans le cas d’ASURO, cette fonction est exécutée après la mise sous tension. La fonction main () doit exister dans chaque programme!

Après avoir vu les types de données et avoir abordé un peu les fonctions, nous allons nous entraîner sur une petite fonction d'exercice qui doit multiplier deux chiffres à 8 bits et renvoyer le résultat.

```
int Mult(char a, char b)
/* la fonction renvoie une valeur int, porte le nom de Mult et reçoit deux char comme
   paramètres  */
{                       // Début de la fonction
        int c;          // Variable c est déclaré comme int
        c = a * b;      // calcule c
        return c;       // renvoie c
}                       // fin de la fonction
```

Et voici encore une petite routine qui exécute la fonction que nous venons de définir:

```
int main (void)                 // La fonction main renvoie toujours un int,
                                // et ne reçoit pas de paramètres
{                               // Début de la fonction
        char mult1,mult2;       // Définition de deux variables char
        int erg;                // Définition d'une variable int qui doit contenir
                                // le résultat de la multiplication des
                                // variables mult1 et mult2l
        mult1 = 2;              // Affectation
        mult2 = 10;             // Affectation
        erg = Mult(mult1,mult2); // Rappel de la fonction Mult définie auparavant
                                   return 0;
}                               // Fin
```

9.1.7. Pointeurs et Vecteurs

Nous ne traitons ici des pointeurs et vecteurs que dans la mesure où ils sont nécessaires pour le fonctionnement d'ASURO.

Si les détecteurs de tracé ou les détecteurs odométriques sont lus, nous avons besoin de vecteurs. Leur déclaration est extrêmement simple :

```
int lData[2];
int oData[2];
```

Il en ressort que 2 vecteurs (I Data, o Data) contenant 2 éléments sont créés pour les détecteurs de tracé ou détecteurs odométriques. Après appel de la fonction correspondante d'ASURO (LineData(), OdemetrieData()), l'élément [0] contient la valeur du détecteur gauche et l'élément [1] celle du détecteur droit.

Voici un petit exemple:
Si sur les deux détecteurs de tracé celui de droite est plus éclairé que celui de gauche, il faut exécuter l'expression 1, sinon l'expression 2.

```
int lData[2];            // Affecter la mémoire pour les valeurs mesurées
LineData(lData);         // Saisie des valeurs mesurées
if (lData[1] > lData[0])
        Expression1;
else
        Expression2;
```

Afin de pouvoir utiliser les fonctions d'interface sérielle (SerWrite(), SerRead()) nous avons besoin de chaînes de caractères qui sont déclarées de la manière suivante :

```
char message [] = "Mettre un texte ici";
```

Pour envoyer une chaîne de caractères dans le cas d'ASURO, il suffit d'appeler juste la fonction SerWrite() avec les paramètres correspondants. Le premier paramètre indique le texte ou bien la variable de la chaîne de caractères et le deuxième paramètre le nombre de caractères de la chaîne qui doivent être envoyés.

```
SerWrite(message,20);
```

Ou bien

```
SerWrite("Mettre un texte ici",20);
```

Envoyer par le biais de l'interface série IR "Mettre un texte ici".

La fonction SerRead () a été définie pour recevoir des caractères dans ASURO. Le premier paramètre contient la variable de la chaîne de caractères dans laquelle les caractères reçus sont mémorisés, le deuxième paramètre indique le nombre de caractères à recevoir et le troisième représente un Timeout. Cela signifie que la fonction s‘arrête si aucune donnée n’a été reçue pendant un lapse de temps défini (temps du processeur). Si le paramètre est réglé sur 0, la fonction attend la réception de tous les caractères.

Voici un exemple:
ASURO doit recevoir par le biais de l’interface IR “Salut me voici”:

```
char message [] = ”01234567890123456789”;
```

L’espace pour recevoir le texte a été créé. La chaîne de caractères créée doit être suffisamment grande pour pouvoir accueillir le texte.

```
SerRead(message,18,0);
```

Enregistre 18 caractères et attend jusqu’à ce que tous les caractères sont arrivés. Nous partons du principe que la chaîne de caractères« Salut me voici » est envoyée. Le message de la chaîne de caractères préalablement défini se présente maintenant comme ceci :

```
Salut me voici456789
```

Les premiers 18 caractères du message ont été écrasés par les caractères reçus.

9.2. Description des fonctions d’ASURO

Afin de faciliter le plus possible la programmation d’ASURO, il existe quelques fonctions prédéfinies. Elles ne sont pas forcément optimales et pour certaines applications il serait certes mieux d’écrire vos propres fonctions.

Les fonctions sont représentées d’une manière classique dans le style de leur déclaration. Celui qui se sent un peu perdu, devrait regarder les exemples.

Afin d’éviter tout malentendu, les fonctions qui commandent quelque chose telles que la fonction d’entraînement ou les fonctions concernant les éléments d’indication fixent des réglages qui restent valables jusqu’à ce qu’elles soient modifiées. Donc, une LED d’état verte reste verte jusqu’à ce qu’une autre couleur soit réglée ou qu’elle est éteinte.

9.2.1. void Init(void)

Le microprocesseur doit être mis dans sa position de départ. Cette fonction doit toujours être exécutée au début d'un programme sinon le processeur ne saurait même pas ce qu'il doit faire de ses jambes.

Un programme pour ASURO doit se présenter au minimum de cette façon:

```
#include “asuro.h”
int main(void) {
                    // déclarer ici les variables requises
        Init();
                    // mettre ici vos propres idées de programme
        while(1);   // Boucle sans fin
        return 0;   //n’est plus exécutée
}
```

Pourquoi la boucle sans fin à la fin de la fonction main ()? Normalement la fin de la fonction main() par return 0 ; signifie la fin d'un programme. Cependant, avec ASURO il peut arriver que des parties d'un programme flashé autrefois soient exécutées ou que le programme démarre à nouveau ce qui entraîne des effets bizarres.
Pour éviter ceci, le programme est « emprisonné » dans une boucle sans fin après avoir effectué toutes ses tâches ce qui représente une fin définie du programme.

9.2.2. void StatusLED(uncaractèred char color)

La LED d'état (D12) peut s'allumer. Les paramètres possibles sont
OFF, GREEN, RED ou YELLOW

Exemple:
La LED d'état doit s'allumer en rouge

```
StatusLED(RED);
```

Okay, okay, encore une fois tout le programme:

```
#include “asuro.h”
        int main(void) {
        Init();
        StatusLED(YELLOW);
        while(1); // Boucle sans fin
        return 0;
}
```

9.2.3. void FrontLED(uncaractèred char status)

La LED frontale (D11) peut être allumée ou éteinte. Les paramètres possibles sont :
ON ou OFF.

Exemple:
La LED frontale soit s'allumer:

```
FrontLED(ON);
```

9.2.4. void BackLED(uncaractèred char left, uncaractèred char right)

Les LED arrières (D15 et D16) peuvent être allumées ou éteintes. Le premier paramètre décrit l'état de la LED arrière gauche (D15) et le deuxième paramètre l'état de la LED arrière droite (D16). Les paramètres possibles sont : ON ou OFF.

Exemple:
Vous voulez allumer la LED arrière droite (D16) et éteindre celle de gauche (D15):

```
BackLED(OFF,ON);
```

9.2.5. void Sleep(uncaractèred char time72kHz)

Cette fonction met le processeur en attente pendant une durée réglée ce qui est idéal pour programmer des temporisations. Elle est basée sur un timer de 72kHz et peut recevoir comme paramètre maximal la valeur de 255 (unsigned char)[4].

Exemple:
Le processeur doit attendre pendant env. 3ms ==> $\frac{0{,}003s}{\frac{1}{72KHz}}$ = 216. La fonction Sleep () e 3ms appelée comme suit pour une attente de 3ms:

```
Sleep (216) ;
```

9.2.6. void MotorDir(unsigned char left_dir, unsigned char right_dir)

Cette fonction détermine le sens de rotation des deux moteurs. Elle doit être exécutée avant le réglage de vitesse. Des paramètres possibles sont FWD (avance), RWD (recul), BREAK (freins ou arrêt, les deux moteurs sont court-circuités par les ponts de transistors) et FREE (roue libre).

Exemple:
Le moteur gauche doit tourner en avant et le moteur droit doit être arrêté :

```
MotorDir(FWD,BREAK);
```

---

[4] C'est de la mauvaise foi de la part des auteurs pour vous forcer à réfléchir!

9.2.7. void MotorSpeed(unsigned char left_speed, unsigned char right_speed)

Ceci détermine la vitesse des moteurs. La valeur maximale possible est 255 (unsigned char). Le moteur ne commence à tourner qu'à partir d'une valeur d'env. 60 (dépend fortement de la construction mécanique). La valeur indiquée ne donne en fait que la puissance électrique que le moteur doit recevoir. Le nombre de tours réel qui en résulte, dépend également d'autres facteurs tels que du frottement ou de la pente.



Dès que cette fonction a été utilisée, ASURO peut démarrer. Parfois le résultat de la programmation n'est pas celui qui était prévu. Vous devez donc vous assurer qu'ASURO ne puisse pas se mettre en danger lui-même ou les autres par des manœuvres inattendues.

Exemple:
Le moteur gauche doit tourner à la vitesse maximale et le moteur droit ne doit pas tourner du tout. Le sens de rotation a déjà été donnée par la fonction MotorDir().

```
MotorSpeed (255,0) ;
```

9.2.8. void SerWrite(unsigned char *data, unsigned char length)

Cette fonction transmet les données d'ASURO par le biais de l'interface sérielle IR avec 2400Bit/s, No-Parity, 1 StopBit, NoFlowControl. C'est le même réglage qui est utilisé pour tester le transmetteur IR (qui l'aurait deviné !). Le premier paramètre contient l'adresse des données à envoyer et le deuxième indique le nombre de bytes à transmettre.

Exemple:
La chaîne de caractères „Salut toi!“ doit être envoyée par l'interface sérielle IR.

```
SerWrite(“Salut toi!”,12);
```

9.2.9. void SerRead(unsigned char *data, unsigned char length, unsigned int timeout)

C'est très bien de pouvoir envoyer des données par l'interface série IR mais il peut aussi être utile d'en recevoir et c'est justement ce que fait cette fonction. Le premier paramètre est le pointeur sur l'espace mémoire où il faut enregistrer les données reçues. Le deuxième paramètre indique le nombre de bytes de données à attendre. Le troisième et dernier paramètre règle la durée (Timeout). Cela évite que la fonction attende des données pendant une durée illimitée. En l'absence de données au bout d'un certain lapse de temps, la fonction est tout simplement interrompue. Le tout premier caractère des données reçues est dans ce cas écrasé par 'T' (Timeout). Si vous mettez « 0 » dans le troisième paramètre, la fonction attend jusqu'à ce que le nombre de bytes indiqués dans le deuxième paramètre, ait été reçu.

Exemple:
Vous voulez recevoir la chaîne de caractères „Demarre“ tout en vous assurant que tous les caractères ont été reçus par ASURO avant de continuer. ***Remarque: Seuls 7 caractères seront reçus. Il n’y a pas de vérification si la chaîne « Demarre » est vraiment arrivée.***

```
#include "asuro.h"
int main(void) {
        char daten[7]; //affecter la mémoire
        Init();
        SerRead(daten,7,0); // lire les données
        MotorDir(FWD,FWD);
        MotorSpeed(120,120);
        while(1); // Boucle sans fin
        return 0;
}
.
.
```

9.2.10. void LineData(unsigned int *data)

Ceci permet la lecture des phototransistors en dessous d’ASURO. Il faut donner une adresse de mémoire qui peut accueillir deux entiers relatifs. Cette fonction remplit alors le contenu de l’adresse avec les valeurs du convertisseur A/D des deux phototransistors. Le premier entier relatif reçoit la valeur de conversion du phototransistor gauche (T9) et le deuxième la valeur du phototransistor droit (T10). La luminosité maximale correspond à une valeur de ‘1023 ‘. L’obscurité correspond à une valeur de ‘0’. Généralement, les deux valeurs extrêmes ne sont jamais atteintes. La valeur mesurée se situe quelque part entre les deux.

Exemple:
Lecture des phototransistors (T9, T10)

```
unsigned int data[2];        //Allocation de la mémoire
.
.
LineData(data);
```

data[0] contient la valeur du phototransistor gauche (T9).

data[1] contient la valeur du phototransistor droit (T10).

Et voici le programme entier:

```
#include "asuro.h"                        // Suivi du tracé de la façon la plus simple
int main(void) {
        uncaractèred int data[2];         //Allocation mémoire
        Init();
        FrontLED(ON);                     // Mise en service éclairage du tracé
        MotorDir(FWD,FWD);                // Les deux moteurs en avant
        while(1){                         // Boucle sans fil; ASURO doit suivre un tracé
                                          // pendant une durée indéfinie
                LineData(data);           // Lire les valeurs de luminosité actuelle des
                                          phototransistors
                                          //
                if (data [0] > data [1] ) // à gauche plus lumineux qu'à droite...
                {MotorSpeed(200,150);}    // ... accélérer plus à gauche...
        else
                {MotorSpeed(150,200);}    // ... sinon accélérer plus à droite!
        }
        return 0;
}
```

9.2.11. void OdometrieData(unsigned int *data)

La barrière lumineuse est évaluée. Les diodes électroluminescentes (D13, D14) sont activées et les valeurs du convertisseur A/D des phototransistors (T11, T12) sont renvoyées. Tout comme dans la fonction LineData(), il faut affecter une mémoire avec deux entiers relatifs qui sera ensuite remplie par la fonction. Le premier entier relatif contient la valeur du convertisseur du phototransistor gauche (T11) et la seconde la valeur intégrale du phototransistor droit (T12). La luminosité maximale correspond à une valeur de '1023 ' [5]. L'obscurité correspond à une valeur de '0'. Généralement, les deux valeurs extrêmes ne sont jamais atteintes. La valeur mesurée se situe quelque part entre les deux.

Exemple:
Lecture de la barrière lumineuse

```
unsigned int data[2];         //Allocation mémoire
.
.
OdometrieData(data);
```

data[0] contient la valeur du phototransistor gauche (T11)
data[1] contient la valeur du phototransistor droit (T12)

Afin d'éviter tout malentendu: OdometrieData() ne lit pas directement le nombre de tours mais seulement la luminosité actuelle du disque de la barrière lumineuse. Il appartient au programmeur d'interpréter les valeurs de luminosité, le comptage des passages clair-foncé et la détermination du nombre de tours de la roue!

---

[5] Pour des raisons techniques, les valeurs sont exactement l'inverse de celles des phototransistors du suivi du tracé. Une concession faite à la simplicité du circuit.

9.2.12. unsigned char PollSwitch(void)

Les capteurs tactiles (K1-K6) sont évalués. Cette fonction delivre un byte. Ce byte nous dit quel capteur tactile a été touché. Le capteur 1 pose le bit 5 et le capteur 6 le bit 0.

Bit.
Bit0 (1) -> K6
Bit1 (2) -> K5
Bit2 (4) -> K4
Bit3 (8) -> K3
Bit4 (16) -> K2
Bit5 (32) -> K1

Si les capteurs 1,3 et 5 étaient touchés, la fonction renverrait 32 + 8 + 2 = 42.

Il faut éventuellement appeler la fonction plusieurs fois à la suite avant d’obtenir le „bon“ résultat.
Le condensateur chargé C7 doit d’abord se décharger et cela peut prendre un certain temps.
Si le convertisseur A/D scanne trop tôt, des valeurs de tension les plus disparates peuvent être mesurées.

Exemple:

```
unsigned char taste;
.
.
taste = PollSwitch();
if (taste>0) {MotorSpeed(0,0);}
```

Voilà, c’est tout.
Maintenant vous pouvez laisser libre cours à votre propre créativité.

Teil IV. Annexes

A. Nomenclature:

En plus d’une balle de ping-pong, il vous faut les pièces suivantes pour la construction d’un ASURO:

1x Platine ASURO
2x Moteurs Type Igarashi 2025-02
1x Diode 1N4001
8x Diodes 1N4148
4x Transistors BC 327/40 ou BC 328/40
4x Transistors BC 337/40 ou BC 338/40
1x Circuit intégré CD 4081BE
1x Processeur ATmega 8L-8PC (préprogrammé)
1x Récepteur IR SFH 5110-36
1x LED IR SFH415-U
2x Phototransistors SFH300
3x LEDs 5mm rouge clair diffus
1x Double LED 3mm rouge/verte
2x Phototransistors Side- LPT80A
2x LEDs Side IRL80A
1x Oscillateur 8MHz
2x Condensateurs électrolytiques 220μ F min. 10V RM 3,5/10
4x Condensateurs céramiques 100nF RM 5,08
2x Condensateurs céramiques 4,7nF RM 2,54
1x 100Ω 1/4 W 5%
2x 220 Ω 1/4 W 5%
4x 470 Ω 1/4 W 5%
10x 1KΩ 1/4 W 5%
1x 1KΩ 1/4 W 1%
3x 2KΩ 1/4 W 1%
2x 4,7KΩ 1/4 W 5%
1x 8,2KΩ 1/4 W 1%
1x 10KΩ 1/4 W 1%
1x 12KΩ 1/4 W 1%
1x 16KΩ 1/4 W 1%
1x 20KΩ 1/4 W 5%
1x 33KΩ 1/4 W 1%
1x 68KΩ 1/4 W 1%
1x 1M Ω 1/4 W 5%
3x Socle 14 br.
6x Détecteurs tactiles
1x Interrupteur
1x Support de pile
1x Jumper
1x Barette 2pol RM 2.5
2x Engrenages 10/50 dents; Perçage 3,1mm Module 0,5
2x Engrenages 12/50 dents; Perçage 3,1mm Module 0,5
2x Pignons moteur 10 dents
2x Bague de réglage pour essieu 3mm

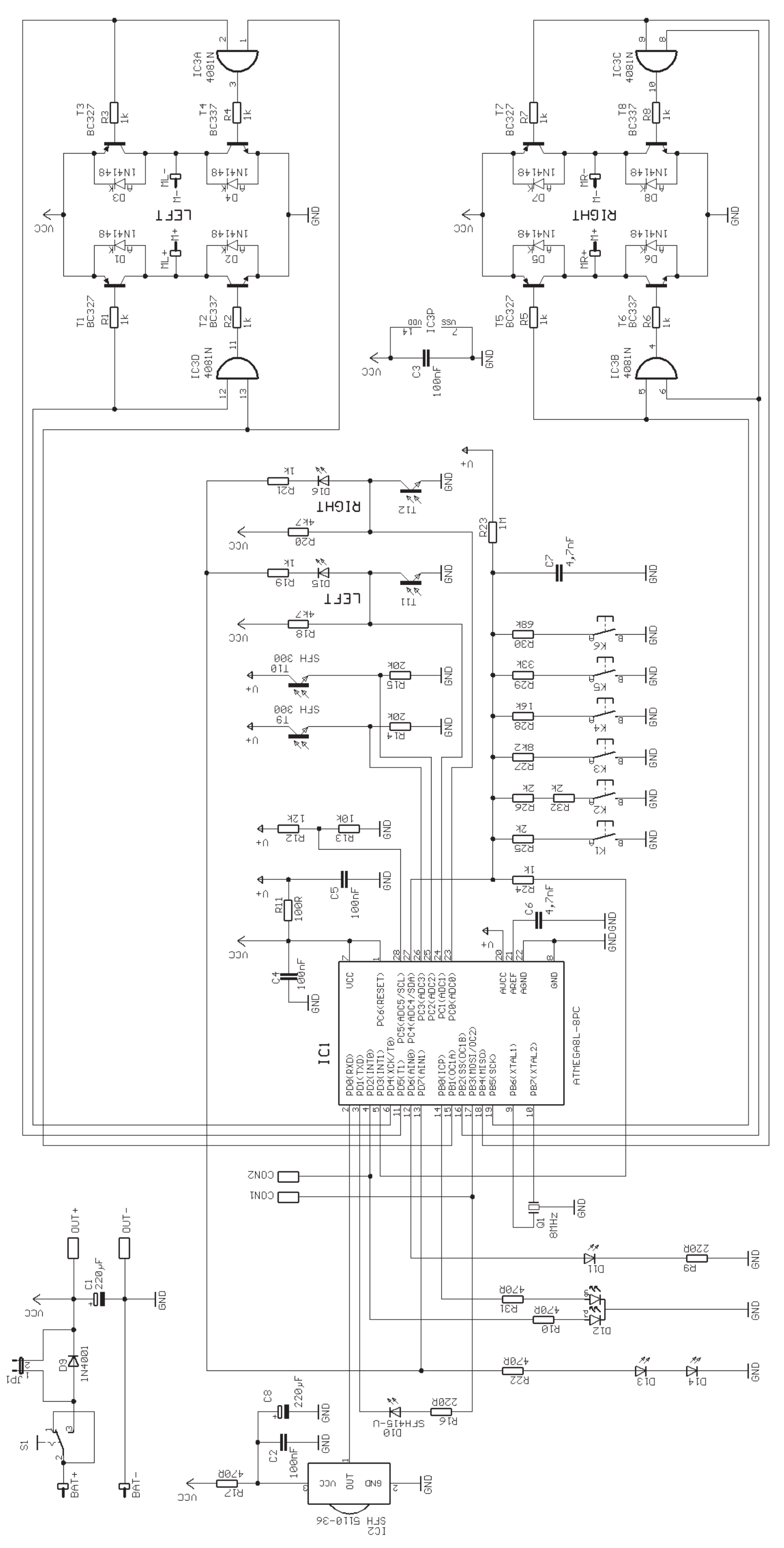
4x Colliers de câble
1x Colliers de câbles amovibles
2x Pneus en caoutchouc 38mm
2x Tige en laiton 42mm de longueur, 3mm de diamètre
2x Tige en laiton 24,5mm de longueur, 3mm de diamètre
Env. 15cm fil multibrins rouge 0,14mm
Env. 15cm fil multibrins noir 0,14mm
2x Disques encodeur (voir 2.4)

Pour le transmetteur IR RS-232, il vous faut les composants suivants:

1x Platine transmetteur IR-RS232
3x Diodes 1N4148
1x Diode zéner ZPD5.1
1x Transistor BC547 A, B ou C ou BC548 A, B ou C
1x Circuit intégré NE555N
1x Récepteur IR SFH 5110-36
1x LED IR SFH415-U
1x Condensateur électrolytique 100 µF au moins 16V RM 2,5/6
2x Condensateurs céramiques 100nF RM 5,08
1x Condensateur céramique 680pF RM 2,54
1x 220Ω 1/4 W 5% ou mieux
1x 470 Ω 1/4 W 5% ou mieux
1x 4,7KΩ 1/4 W 5% ou mieux
1x 10KΩ 1/4 W 5%
2x 20KΩ 1/4 W 5% ou mieux
1x Trimmer 10KΩ droit RM 2,5/5
1x Socle 8 pol.
1x Fiche SUB-D 9-broches

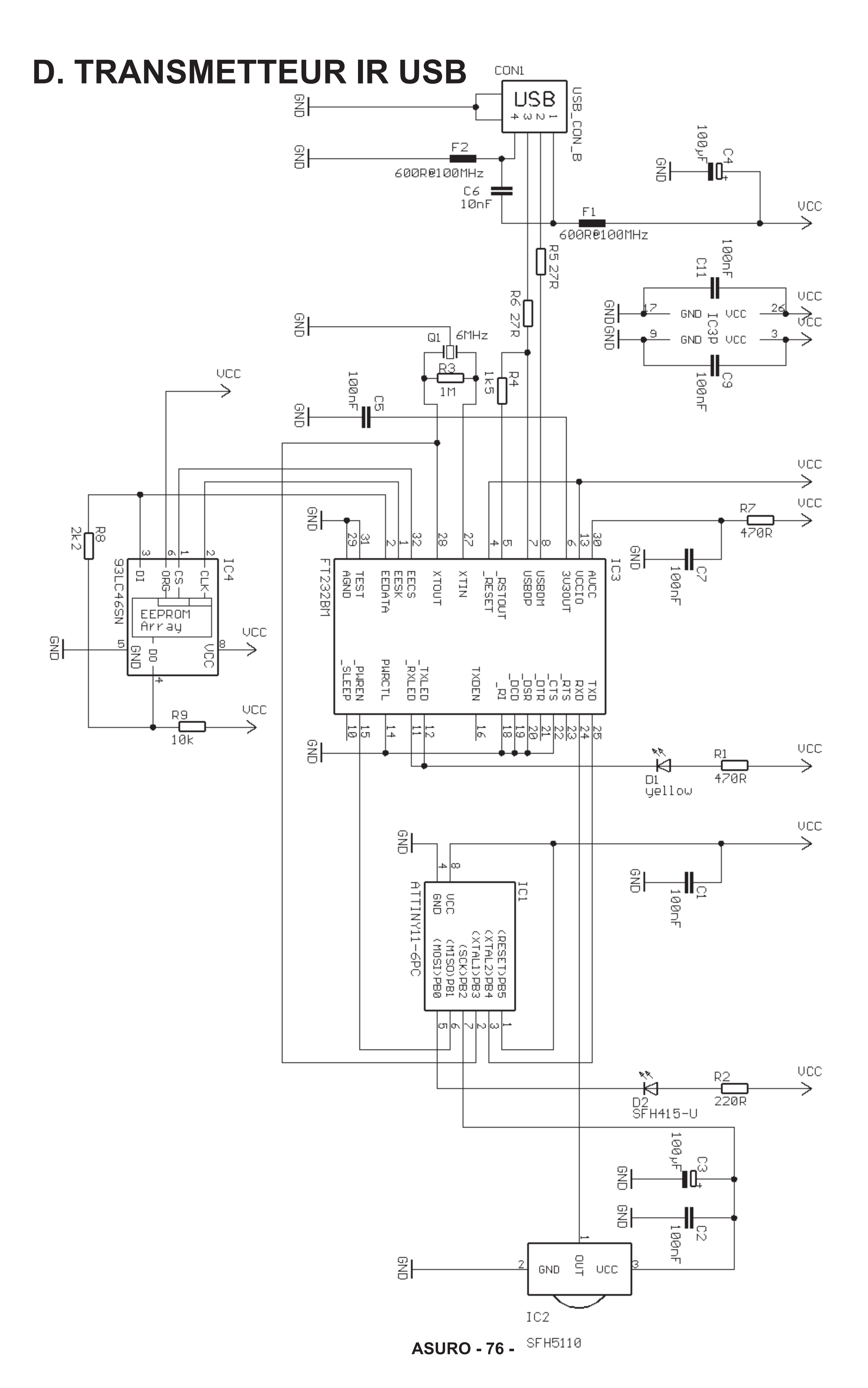
Vous avez la possibilité d'utiliser le transmetteur USB:
Le transmetteur IR USB est également livré comme produit fini !

B. Schémas techniques ASURO



C. TRANSMETTEUR IR RS232

D. TRANSMETTEUR IR USB



E. SYNOPTIQUE ASURO

Transmetteur Infrarouge RS232/USB 2400 Bd

l'interface série (Communication) (Programmation)

Vérification de la pile

LED d'état (Bicolore)

Microcontroller (Atmel ATmega8)

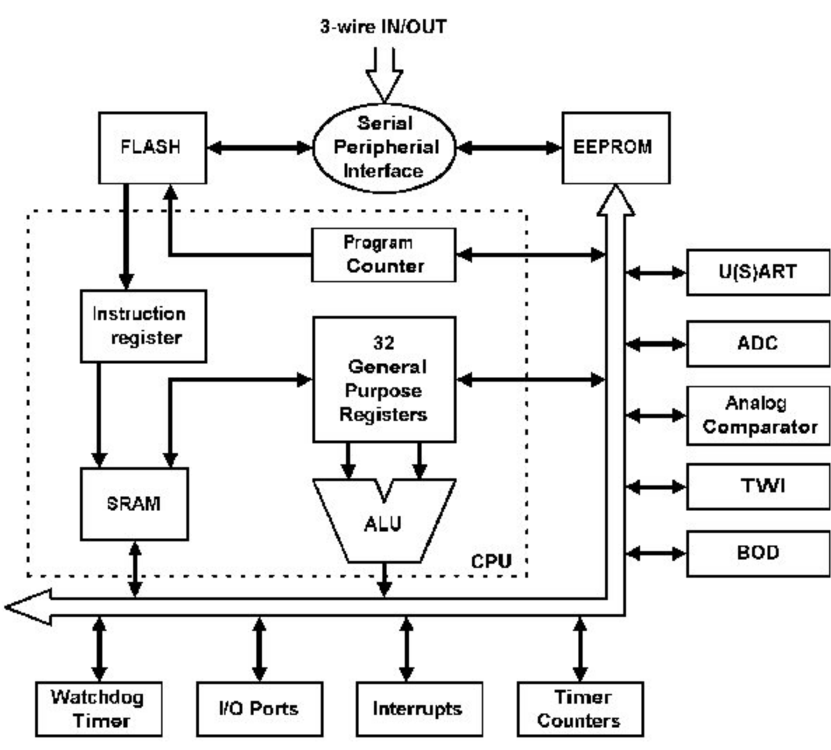
LED arriére

Entraînements

Détecteurs de tracé (Phototransistor, LED)

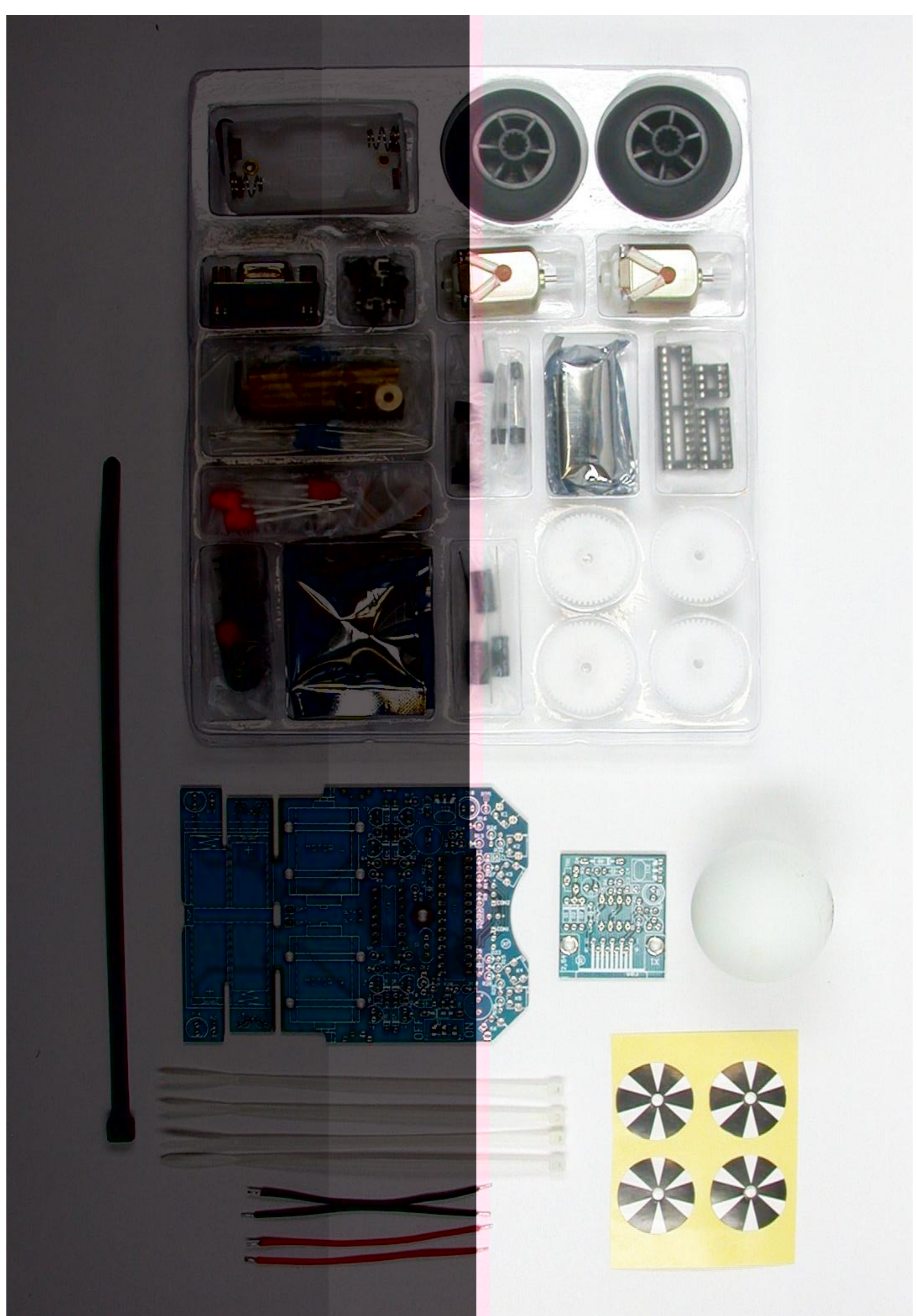
Odométrie (Barrière lumineuse réfléchissante)

Détecteurs tactiles (6 Commutateurs)



F. SYNOPTIQUE PROCESSEUR AVR

G. CONTENU D'ASURO



Informatique la partie électronique

Lorsque le composant comporte des repérages à proximité des pattes, la polarité est extrêmement importante !
Une polarité erronée peut endommager le composant voire le circuit complet !
Pour plus d'information voyez aussi le Fiche Technique Dossier!

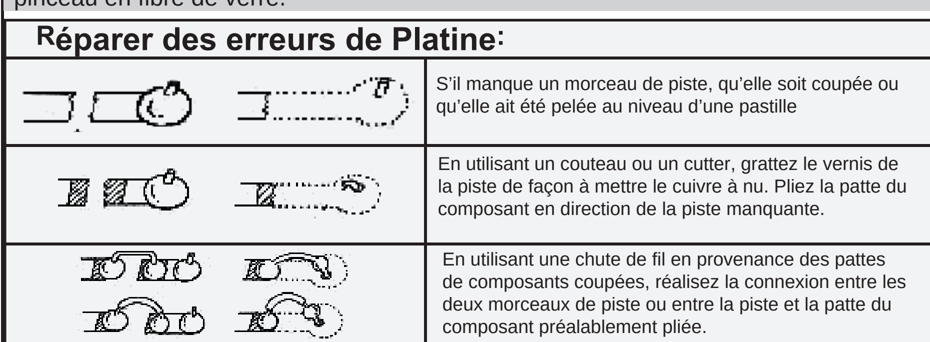
Composant	Montage	PCB symbole	Circuit symbole
Résistance variable **(Potentiometer)**			
Résistance			
Condensateur			
Condensateur électronique (ELCO) - +			
Transistor		*B et E sont simplement des exemples, la position peut varier en fonction du type* B E	*PNP type* *NPN type*
(Zener)Diode *A* *C*		*A* *C*	*A* *C*

Composant	Montage	PCB symbole	Circuit symbole
LED rouge D15-D16 A *Rouge*	*C (Cathode) = fil court sur la face repérée*	*fil court = C = -*	
DUO LED D12	*le fil le plus court doit aller dans le tracé carré*		
IR-LED IRL80A Side type Transparent	***Notice !*** *Respectez la bonne polarité*		
IR-LED SFH415-U *Noir*	C A C A *C (Cathode) = fil court sur la face repérée*		
IR-Receiver SFH5110	***Notice!*** ***Côté bombée vers le haut***		
PHOTOTRANSISTOR SFH-300 *E* *Incolore*	*Respectez la bonne polarité*		
PHOTOTRANSISTOR Side type LPT80A Rose colore	***Notice!*** *Respectez la bonne polarité*		
Ceramic Oscillateur Q1	***Non polarité***		
IC Atmega 8L	*Respectez la bonne polarité*		
IC CD4081	*Respectez la bonne polarité*		
IC NE555	*Respectez la bonne polarité*		

Composant	Montage	PCB symbole	Circuit symbole
Connecteur 2 broches JP1			
Capteurs K1-K6			
Interrupteur (On/Off) S1			
IC (Socket)			NON SYMBOLE
Fiche SUB-D connecteur			
Moteur			NON SYMBOLE (M+ Gauche/Droit M-)

Réparer des erreurs de soudure

Les circuits imprimés sont constitués d'une piste de cuivre qui établit une liaison électrique avec les pattes de connexion des composants. Le cuivre est recouvert de soudure aux pattes. Une couche isolante de vernis vert protège la piste de cuivre contre des courts-circuits et l'oxydation (formation de rouille). Si nous chauffons le cuivre trop longtemps lors de la soudure, la patte et la piste de cuivre peuvent se détacher de la plaquette. Afin de réparer les dommages, nous sommes souvent obligés d'enlever une partie de la couche de vernis sans égratigner la piste de cuivre. Pour cela, il est recommandé d'utiliser un couteau avec une lame courbée (p.ex. un cutter) ou en pinceau en fibre de verre.



Réparer des erreurs de Platine:	
	S'il manque un morceau de piste, qu'elle soit coupée ou qu'elle ait été pelée au niveau d'une pastille
	En utilisant un couteau ou un cutter, grattez le vernis de la piste de façon à mettre le cuivre à nu. Pliez la patte du composant en direction de la piste manquante.
	En utilisant une chute de fil en provenance des pattes de composants coupées, réalisez la connexion entre les deux morceaux de piste ou entre la piste et la patte du composant préalablement pliée.

